Agent-Based Robot Control Design for Multi-Robot Cooperation

Chia-How Lin and Kai-Tai Song
Department of Electrical and Control Engineering
National Chiao Tung University
Hsinchu 300, Taiwan R.O.C.
jotarun.ece87@ntcu.edu.tw, ktsong@mail.ntcu.edu.tw

Gary T. Anderson
Department of Applied Science
University of Arkansas at Little Rock
Little Rock, AR, U.S.A.
gtanderson@ualr.edu

Abstract - This paper presents an agent-based robot control (ARC) architecture. ARC features a flexible real-time control system, which is suitable for multi-robot cooperative tasks. It also provides an efficient platform for building up a multi-robot system consisting of heterogeneous robots. In this paper, an experimental study of this architecture is investigated. A cooperative exploration using two mobile robots will be demonstrated. In this experiment, one robot explores the environment by looking for a color-coded target and the other is responsible for task execution at the target position. While exploring in an unknown environment, the first robot, which is equipped with ultrasonic sensors for exploration, records its position as it sees deployed checkpoints. In a later phase, the second robot plans a path to the target directly using information passed from the first robot and get to the target position in an efficient way.

Keywords: mobile robots, agent-oriented programming, multi-robot cooperation, robot exploration

1 Introduction

For many complex tasks in service robotics, such as cleaning or load transport, it is desirable to take advantage of the flexibility provided by using multiple robots. It would also be interesting to adopt a multi-robot approach to assisting disaster rescue, such as if an earthquake or fire occurs. A multi-robot cooperative system appears to be more effective and adaptive to accomplish various complex tasks, relative to a single robot approach. The complexity of each robot can be reduced to minimum, which means lower costs, simpler design, and faster computation [1]. The execution of specific tasks, e.g. exploration, can be more efficient and robust since many robots can work collaboratively [2, 3]. Moreover, the system can be altered easily to adapt to different circumstances and missions, by forming teams with robots that have the necessary abilities. These benefits, however, often come with drawbacks such as increased complexity in software for control and coordination of multiple robots, the need for inter-robot communications, etc.

A flexible and powerful control architecture are therefore desirable to provide a structured approach to simplify and organize the design, implementation and validation of such a complex control system as well as its subsystems. The multi-robot control system ACTRESS proposed by Asama et al. [4] is an excellent example of such concept. In their architecture, it is necessary to access to the state of a multi-robot system, which is required for decision-making and control. Also, it is needed for coordination of the activities of individual robots to accomplish a desired team activity.

In recent years, multi-agent approaches to multi-robot control design have drawn much attention in the robotics community. In an agent-based architecture, agents can communicate, coordinate and negotiate to meet their goals in a framework suitable for task execution. The architecture supports both centralized (agent facilitator model) and distributed (agent to agent model) arrangement. For example, NASA’s CAMPOUT [5] is a control architecture which defines the abstract design of a class of agents: a set of structural components in which perception, reasoning, and action occur; the specific functionality and interface of each component, and the interconnection topology between components. In [6], Vidal et al. address multi-agent control architectures for teams of ground and aerial vehicles. They presented a hybrid hierarchical system that segments the control of each agent into different layers of abstraction. These layers of abstraction allow interoperability in heterogeneous robot teams. They illustrate the effectiveness of this approach in a pursuit-evasion application. A review of other researches can be found in [7], with particular emphasis including task planning, fault tolerance, role assignment, etc.

In order to synchronize the motion of each robot in a multi-robot system, inter-robot communication plays an important role. In most robot control architectures, communication is treated as a lower-level function. However, it would be more general and flexible to handle the communication as a higher level perception, in order to be managed within the framework of whole system. This is especially useful for a multi-robot system, where in practice the communication might encounter problems such as latency or even break off, and the system should be able to handle the problem under a global scope of view. Furthermore, onboard devices and programming representation of components are often different on heterogeneous robots. Thus a high level abstraction such as using middleware agent will be beneficial to solving this problem in system development.

In this paper, an agent-based robot control architecture is proposed for providing a generic procedure for building
a multi-robot system. A design for handling the communication problem will be illustrated. The purpose is to integrate multiple heterogeneous autonomous robots into a coordinated system that is modular, scalable, and efficient.

2 Proposed ARC Architecture

Figure 1 illustrates the basic structure of ARC. The agent-oriented robot control architecture is a distributed hybrid system, which couples reactive and local deliberative agents. ARC includes a multi-agent framework and a software architecture, which provides both flexibility and simplicity for robot control development. In this architecture, Router manages the inside communication of the robot, while other components handle the real-time performance of the robot behaviors. Each individual agent is responsible for sensing, actuating, or executing a set of functions. An agent also exchange information with other agents. ARC is suitable for system that uses different hybrid architecture. Furthermore, different robots can be implemented simply by adding and modifying the reusable agents. It is therefore efficient to establish a group of heterogeneous robots. For detail features and implementation about ARC, refer to [8].

When implementing the agent architecture in a multi-robot control system, communication latency between robots needs to be considered. In practice, wireless communication such as wireless LANs or Bluetooth protocols are often used on robots. Since the hardware varies and the transmission rate is not deterministic, it is difficult to integrate these communication functions into a real-time system. To deal with this problem, transmitters and receivers (or the server and client in internet connection) are treated as one of the sensors and actuators of the robot in the purposed ARC architecture. Each robot therefore contains several pairs of sender and receiver agents, related to different communication devices and protocols (such as http, ftp, etc). Therefore, the robots are able to build a rendezvous system, in which the network can be established dynamically. Moreover, the robots can establish a communication network not only with other robots, but also with devices such as a PDA or a central host computer. Therefore the cooperative teams can be formed with more flexibility under this design. With communication agents, cooperation between multiple robots can either distributed, centralized or even mixed.

The communication agents can fulfill both high level communication such as task assignment and lower level communication such as data exchange. When connections between robots change or break off dynamically and unexpectedly, communication agents can handle and report the problem in order to perform suitable treatment under a global point of view.

Figure 2 illustrates an example of how two robots exchange data. It is clear that this architecture can be extended to handle more than two robots. This is also the software architecture used in the experiment described later.

3 Implementation

3.1 Hardware configuration

The proposed agent-based control system is implemented on two robot platforms (robot 1 and robot 2) constructed in our lab. Figure 3 shows recent pictures of these two robots. The basic structure of both robots is similar. Both are equipped with two independent drive wheels and two casters for balance. The motor servo control is implemented on self-made DSP motion control boards. The onboard PC (Celeron 300MHz) communicates with DSP motion control boards via RS-232 serial link. Each robot uses two 12V batteries, two 6V batteries and a 12V to 5V DC-DC converter for power supply. An USB Webcam installed on a 2-DOF pan-tilt robotic head provides a vision system to the robot. Several IR modules on the robot can receive digital codes from IR transmitters to get the information from external checkpoints. In the cooperative exploration, robot 1 is programmed in particular for exploration, and robot 2 for task execution.
Therefore, robot 1 is equipped with twelve ultrasonic sensors for environment detection, and robot 2 is equipped with a 3-DOF arm for grasping.

3.2 Software configuration

The ARC system has been implemented in C/C++ under Linux (kernel 2.4.18) and RTAI (version 24.1.13) in order to take advantages of their real-time features. The agents designed for the two robots are described in the following sections.

A. Robot localization

In order to find an efficient path and to follow it in a later phase, the robots need a localization method to estimate their pose in a 2D plane. For short traveling distances, odometry from dead-reckoning can be used to estimate the pose of the robot based on counting the encoder pulses from wheel shafts. However, this method suffers from accumulated error over distance traveled. Therefore, some artificial devices termed checkpoints as shown in Figure 4 are placed in the environment as beacons. The checkpoint provides absolute coordinate information, allowing robots to periodically recalibrate their positions, thus reducing odometry errors.

A checkpoint consists of two major parts:

(i) Four IR emitters, each of them transmits a different digital code for encoding local information of the work space. The main function of this part of checkpoints is to provide an absolute coordinate of the robot nearby individual checkpoint. Since the absolute coordinates of each checkpoint are known in advance, the location of the robot can be estimated as its relative posture to the checkpoint is known. This information can be used to correct the accumulated position errors caused by wheel slippage in dead reckoning.

(ii) A specific visual pattern, which is formed by a tube with two colors as shown in Fig. 4. The distance from the robot to the checkpoint can be estimated based on the size of each part of the tube in the image plane as taken by a camera onboard the robot.

Note in the lower right of Fig. 4, four IR transmitters are placed respectively at 0°, 90°, 180°, and 270° around each checkpoint, and each transmitter has its own digital code that indicates its unique position and orientation. A microcontroller is used to realize the control and data encoding/decoding of IR signals for both the transmitting and receiving modules. The robot control computer receives the transmitted data using the receiver module onboard via a serial port to obtain specific information from the checkpoint. When a robot receives an IR signal with a specific digital code, the robot can determine its pose and position. The valid range of the IR sensor is from 0 to 2 meters.

In each robot controller, a checkpoint sensor agent is responsible for this procedure. It obtains information from the image sensor agent and IR sensor agent, and then updates the recorded coordinates of the robot. Furthermore, this agent is able to locate the target, which is a special color pattern placed in the environment. We simply implemented with the same tube used for the checkpoint but with two colors reversed (i.e. the checkpoint color tube upside down). In the experiment, robot 1 first explores the environment and locates the path to the target. The observed information is transferred to robot 2, which can then navigate directly to the target object.

B. Programming the Exploration Robot (Robot 1)

The programming organization of robot 1 is depicted in Figure 5. The wall following agent directs the robot to explore the area along a wall, while the safety monitor agent avoids obstacles using data from the ultrasonic sensors. During exploration, a map builder records the coordinates of the robot, which are updated by the odometer and checkpoint sensor agent. When the target is found, the map builder constructs an optimal path and then
transmits this information to robot 2 by means of the wireless LAN as well as the communication agent. The
following steps are implemented in robot 1 to generate a path:

(i) Step 1: Detect if there are any checkpoints, if a check point is found, then record its coordinates

(ii) Step 2: Mark a virtual via (target) point in front of the detected checkpoint (the final target object). These via-points are used to construct the desired route for robot 2 to follow to reach the target position.

(iii) Step 3: Plan an optimal path to the target based on the via-points.

Meanwhile, the decision maker of robot 1 will launch a command sender to ask robot 2 to go to the target, and transmit the optimal path to reach the target. For the current design, Robot 1 will wait for the arrival of robot 2. If robot 2 confirms the target, the command receiver of robot 1 will read a response from robot 2 and robot 1 resume exploring the rest of the area.

C. Programming the Task Execution Robot (Robot 2)

The programming organization of robot 2 is illustrated in Figure 6. Several agents programmed for robot 1, such as the checkpoint sensor, command sender, etc, are reused for robot 2. A target achieving agent is added to provide the ability to go from one target to another.

The execution sequence in this case is as follows:

(i) Update the coordinates of the robot based on the data from the checkpoint sensor agent and the odometer.

(ii) Calculate the distance between the robot and the via-point or target point.

(iii) Calculate the angle between the robot and the via-point or target point.

In the experiments, in order to obtain a smooth motion of the mobile robot, position and orientation correction are performed conditionally at every checkpoint. If the angular error is greater than 10 degrees, the robot will then try to compensate the orientational error by a rotation. Otherwise, it will continue to approach to the target or via-point. This rule is put to work in order to avoid motion fluctuations.

Once the route data and the commands from robot 1 are received, robot 2 will execute the target-achieving agent and start travel from one via-point to the next ones. When robot 2 arrives at the final target, it will switch control to an external control module in order to execute its special actuation function (e.g., grasping). Meanwhile, robot 2 will inform robot 1 to resume exploring the environment.

4 Experimental Results

4.1 Goal

An experiment study has been designed and carried out to show how multiple robots with different abilities can cooperate to achieve an assigned task without a central coordinator. Participating robots share their information obtained individually and finish the task according to their specific skills. Two heterogeneous robots are used in the experiment. One robot equipped with sonar sensors is responsible for exploring the area and finding the path to the desired target. The other robot, equipped with a three degree-of-freedom robotic arm and gripper will be in charge of physically acquiring a target object, while following the path transmitted by the first robot in order to efficiently reach the object. The capability of the robots to effectively cooperate will be tested in this experiment.

The experiment was carried out in the corridor of our lab building. Figure 7 shows the experimental setting and the recorded trajectories of the two robots. A local coordinate system is overlaid on the setting for data analysis purposes. The target is located in an open space near a corner of corridor, with a coordinate in meter of (12.8,-3). Four checkpoints were placed at the coordinates, (1.2, 1.2), (6, 1.2), (10.8, 1.2), and (15.6, 1.2), respectively. The starting location of robot 1 and robot 2 are (0, 0) and (-4, 0) respectively, as shown in Figure 8(a).

The execution of this experiment is divided into two stages:

A. Stage 1: Exploration of robot 1:

Since the environment was unknown in the beginning of the experiment, robot 1 explored the environment using sonar sensors. It first moved forward, since there were no obstacles in the corridor. Figure 7 shows that the recorded trajectory of robot 1 is a straight forward motion. However, the actual path of robot 1 is slanting to the right. This error was corrected every time robot 1 came to a checkpoint. These corrections caused a sudden change in the recorded trajectory, which can be observed in Figure 7.

When robot 1 traveled nearby the third checkpoint, the sonar sensor agent detected a free space on its right. Robot 1 therefore made a right turn until it found a wall to follow again. This phenomenon can also be observed in Figure 7, at the coordinate (-0.5, 11). Later, robot 1 found the target object with the onboard webcam and then stopped. While exploring, robot 1 marked each via-point at 1.6 m in front of every checkpoint, and the target point at 0.3m to the left of its location when it found the target. The coordinates of these points are depicted in Figure 7. Robot 1 then sent this information to robot 2, and requesting it to come to the target to perform the task. Fig. 8 shows photos of the experiment.
B. Stage 2: Path tracking

After robot 1 reached the target point, robot 2 started to move along the path transmitted to it by robot 1. Robot 2 first turned and reached via-point 1. It adjusted its coordinates with the aid of the checkpoint, and then continued to move toward via-point 2. Since the angular error was smaller than the threshold (10 degrees), robot 2 did not adjust its pose at this point. When robot 2 came upon via-point 2, it adjusted its position when observed this checkpoint, as shown in Figure 7. After the adjustment, robot 2 found its actual angular error was over the threshold, and therefore changed the heading direction to the next via-point, as shown in Figure 8(b). When robot 2 arrived at the target point (Figure 8(e)), it signaled robot 1 to resume exploring. Figure 8(f) shows that the robot stopped at the final target point as expected.

It was observed in the experiment that the checkpoint and path information collected by robot 1 were correctly transferred to robot 2, which used the information to directly proceed to the target. Note that the accumulated odometry errors during the robot motion were corrected by using the checkpoints. We see from Figure 7 that the executed trajectory is adjusted for robot 2 at all via-points: 1, 2 and 3. Without the position correction at these checkpoints, robot 2 would suffer from the accumulated error, and cannot reach the given via-points and target point as desired. The experimental result in Figure 7 reveals that the robot 2 does use the checkpoints to eliminate accumulated error, and follow via-points and reach the target point accurately.

5 Conclusion and Future Work

This paper presents an agent-based control architecture for multi-robot system. With the properties of the system such as usability and connectivity, one can efficiently build a heterogeneous multi-robot team. Experimental results reveal a basic success of the system architecture. Two
robots were able to explore a room autonomously, share information and work together in a manner that allows the robot with special equipment to exploit its unique capabilities.

In the future, the system will be extended to investigate practical applications, for example, home security. The robots will cooperate with intelligent devices in the environment to gain more information from the environment. A distributed surveillance system can be constructed using the ARC architecture for the whole multi-robot team.

Acknowledgement

The authors thank Chih-Huo Chen and Chi-Yi Tsai for their support in performing the experiments. This work was supported by the National Science Council, Taiwan R.O.C, under grant NSC 92-2213-E-009-007.

References


