A Self-Reconfiguring Team of Mobile Robots (Extended Abstract)

Robert Fitch∗, Alen Alempijevic†, and Ritesh Lal∗

∗Australian Centre for Field Robotics (ACFR)
ARC Centre of Excellence for Autonomous Systems
The University of Sydney
Sydney, NSW Australia
{rfitch, rrlal}@acfr.usyd.edu.au

†Mechatronics and Intelligent Systems Group
ARC Centre of Excellence for Autonomous Systems
University of Technology Sydney
Sydney, NSW Australia
Alen.Alempijevic@eng.uts.edu.au

Abstract—This paper studies coordinated motion planning for a large team of mobile robots communicating over a wireless mesh network. Applying concepts from self-reconfiguring modular robotics, we address connectivity and motion planning in the context of a formation of mobile robots that dynamically adapts to its environment or task. We present a planning and control algorithm that is fully decentralized and assumes local (neighbor-to-neighbor) communication only. We present hardware experiments with a team of nine mobile robots that communicates using a standard ZigBee network. Our results demonstrate useful and complex group behavior in a real robot network, and we discuss properties of our algorithmic approach that are well-suited to properties of real wireless networks.

I. INTRODUCTION

Large networks of mobile robots pose interesting challenges in motion planning and control. The most common approach to tackle these challenges is based on control theory [1]. Another main approach is based on decentralized data fusion (DDF) [2]. We are interested in a third approach, based on planning, that does not require low-latency synchronous communication and is well-suited to implementation in a wireless mesh network such as shown in Fig. 1.

We are interested in the problem of formation reconfiguration, which can be viewed as an extension of problems such as formation control and flocking. The idea is to dynamically change the structure of a formation to adapt to a specific environment or task. The goal formation either can be defined loosely as a geometric bounding region (such as a convex polygon), or can be defined exactly by specifying a list of robot positions. By iteratively choosing goal formations at an offset to the current formation, formation reconfiguration can generate group locomotion.

There are several good reasons to study reconfiguring formations. One is as a complementary problem to formation control. The formation can dynamically change in response to obstacles, such as to squeeze through a small gap. Another motivation is to implement virtual obstacles that constrain the motion of the team to designated areas such as lanes on a roadway or virtual lanes in a manufacturing or cargo handling situation. Alternatively it could be useful to specify the shape of a formation for a specific task, such as a sensing task, without specifying the position of any individual robot. Finally, formation reconfiguration can be used for group navigation. This is particularly useful in the case of many small obstacles or in the human-robot interface context where it is desirable for one user to semiautomatically command n robots.

The two main algorithmic challenges are connectivity preservation and motion planning. The implementation challenge is how to cope with potentially high-latency message delivery. Our approach, an extension of our existing highly scalable algorithm for self-reconfiguration in modular robots [3], is fully decentralized, preserves connectivity using a combination of graph search and locking, and computes a navigation function using decentralized dynamic programming. The main assumptions are that robot positions are structured as a 2D lattice, relative localization is available, neighbor-to-neighbor communication is available, and robots have short-range obstacle sensing capability.

The major benefit of our approach is its efficiency and suitability for implementation in a wireless network; asynchronous dynamic programming does not require low-latency

Fig. 1: Robot hardware used in our experiments: (a) custom electronics subsystems, (b) iRobot Create base with added components for sensing, communication, and control.
message passing. The algorithm is also beneficial in planning around many small obstacles, squeezing through small gaps, and for splitting or merging operations. Further, it is easy to integrate DDF state estimation in planning. For example, the center of the formation is trivially computed using DDF. The convex hull of the formation’s perimeter also is easily tracked using DDF and the merge-hull algorithm.

In this paper, we propose an algorithm for formation reconfiguration and present an initial experiment with real robots. Our experiment validates the approach and shows that hardware implementation is straightforward with minimal computation and memory resources. We discuss implications of random message delays on system performance and outline our plan for ongoing experimental evaluation.

II. RELATED WORK

Recent work by Michael and Kumar [1] addresses the problem of planning for formations with variable shape using distributed control laws with provable guarantees. The key difference is that our approach does not rely on any centralized process to track the current global shape of the formation. Our algorithm never explicitly computes the global shape of the formation and instead uses distributed dynamic programming for parallel path planning.

There is a large volume of research in distributed formation control for teams of mobile robots. A recent survey is by Murray [4]. Other good examples include [5], [6], and [7]. The approach of using a virtual (fixed) structure is presented in [8]. Decentralized navigation functions are investigated in [9] for the problem of stabilizing small (three- or four-robot) regular formations. The problem of maintaining connectivity is studied in [10], among others. Our approach is distinct from this body of work in that it is derived from a planning rather than control-theory perspective.

Many algorithms have been proposed that address the reconfiguration planning problem in self-reconfiguring robots [11], [12]. The algorithm we present in this paper is an extension of our Million Module March algorithm [3].

III. SELF-RECONFIGURATION ALGORITHM

We formulate the formation reconfiguration problem for robot teams in terms of graph reconfiguration. The problem is to transform one formation into another through a sequence of primitive moves. A formation is defined as a graph where each node represents a robot and nodes are embedded in a 2D lattice. Edges connect adjacent nodes. A goal formation is represented geometrically as a bounding region. Other representations are possible, including a complete list of node positions. The goal is generated manually by a user or autonomously by a separate decentralized process. A move is a three-step process whereby a robot 1) removes its node from the graph, 2) physically moves to a new lattice position, and 3) re-inserts itself into the graph. The graph must remain connected following node removal. The main assumptions of our approach are that node position is restricted to a planar lattice and goal shapes do not have holes. Further details and analysis of the algorithm in its original context are presented in [3] and [13].

A. Connectivity

The connectivity check algorithm guarantees that the robot graph remains connected during simultaneous moves. For any given node, the approach is to search for a set of paths in the graph that connect each pair of neighbors. Nodes along these paths are then locked. By definition, this condition is sufficient for preserving graph connectivity. All nodes execute their search asynchronously and in parallel. Locks are non-exclusive and can be shared between multiple moving nodes. Locks are cleared after a node is reinserted in the graph. Deadlock can occur if two nodes attempt to lock each other, but this situation is easily prevented with arbitrary prioritization such as module ID.

We implement the search using message passing between neighbor robots. Search proceeds in an iterative-deepening fashion. Memory requirements scale with the number of simultaneous searches in which a given robot can expect to participate. In dense formations, the search typically succeeds at depth two and memory requirements are minimal.

B. Motion Planning

Our motion planning algorithm uses decentralized dynamic programming to compute a navigation function. This function encodes shortest paths from each robot to its closest position in the goal formation. Because the graph structure continually changes, the navigation function is updated in response to each robot move. This replanning is similar in character to the idea of prioritized sweeping in reinforcement learning.

In a square lattice, two motion primitives are defined. Axis-aligned moves are implemented by the robot as a rotation and an axis-aligned translation. A diagonal move is implemented as a path with constant curvature. A simple PD controller is used to track the desired path.

Dynamic programming updates are implemented with message-passing between neighbors. Updates propagate away from the goal. Empirically, convergence time is proportional to the diameter of the graph. This is a favorable measure as the diameter of a square formation of $n$ robots is $O(\sqrt{n})$.

IV. ROBOT HARDWARE

Our experimental system consists of nine iRobot Create mobile bases controlled by custom electronics, shown earlier in Fig. 1. The Create is a differential-drive base with onboard power and good-quality wheel encoders.

The Create is connected via a serial interface to our custom electronics. The embedded platform design is based on the ST Microelectronics ARM Cortex-M3 CPU, a 32-bit processor with a clock speed of 72MHz, 512kB flash memory, 64kB of RAM, and 12-bit ADC. A micro-SD card slot is included and is intended for storage of large files (currently up to 16GB) such as data logs. For debugging purposes, the platform includes a 1.5”, full-color, 128x128 pixel
Fig. 2: Illustration of experimental setup. Locomotion results from placing the goal formation, shown as a wireframe box, at an offset to the current formation. Robots drawn as circles.

Fig. 3: A sequence of four snapshots from our experimental demonstration. The horizontal white line is a reference that marks the forward edge of the original formation.

OLED (Organic Light Emitting Diode) display. Wireless communication is implemented with off-the-shelf, 2.4GHz ZigBee hardware modules from Digi.

The software system uses a real-time operating system (RTOS), the Crossworks Tasking Library (CTL). CTL is a pre-emptive multitasking RTOS with concurrency support. We developed a custom over-the-air update system that allows us to reprogram software using the wireless communication link [14].

Odometry-based localization is performed by integrating wheel encoder data provided by the Create. The system also supports acoustic localization through a peripheral electronic subsystem that consists of a MEMS microphone (Knowles Acoustics SPM0404HE5H) and digital signal processor (Texas Instruments TMS320VC5505). We will integrate odometry and acoustic localization into a single EKF estimate.

Range sensing is supported by a scanning infrared sensor. An emitter/detector pair is mounted on a stepper motor controlled by the main processor. Range sensing is not used in the current experiments but will be used for obstacle detection in planned experiments.

V. EXPERIMENTS

We implemented the algorithm and performed an initial experiment. The two goals of this experiment were to: 1) validate assumptions on computation, communication, and memory requirements, and 2) validate algorithm performance in terms of number of parallel moves (number of robots moving at the same time) and speed (clock time) of planning. We used our hardware-in-the-loop simulator [14] for development and debugging, where computation and communication are embedded but actuation is simulated, before experimenting with the full system.

The experimental setup is diagrammed in Fig. 2. The robots are arranged in a square lattice and the goal region is defined as a bounding box. The goal region is defined relative to the robots, such that it intersects with at least one robot. The goal is sent by the user via a desktop computer to all robots. This broadcast triggers reconfiguration to begin. Reconfiguration ends when all robots are contained within the goal. The user then broadcasts a new goal region and the process repeats.

A sequence of snapshots is shown in Fig. 3. Results from the test are positive. We observed that up to four robots moved simultaneously, and time required for planning was less than one second on average.

This experiment relied on odometry only for localization and was limited to two iterations of moving the goal region. We are now conducting experiments using an acoustic localization system where robots receive a signal from multiple active beacons and compute their positions by solving time-difference-of-arrival equations. We are performing longer experiments (more iterations) with more complex changes in goal position and shape. We log time-stamped message data and control actions to analyze the performance of the system.

VI. DISCUSSION AND FUTURE WORK

Our experiments demonstrate that the proposed algorithm performs well with a nine-robot team and relatively low-bandwidth mesh network. We know from previous work that the ZigBee network will not support larger teams [13]. However, we are currently developing a custom multi-radio multi-channel system designed for this purpose [15]. The idea of this custom system is to support communication between all pairs of neighbors at high bandwidth, simultaneously, by exploiting the assumption that each robot has
a limited number of physically proximal neighbors. This system should allow us to perform experiments with much larger teams.

We have demonstrated the algorithm’s computational efficiency in previous work, but its benefits in terms of communication have only become apparent when working with real hardware. The communication requirement, although large in terms of number of small messages, is asynchronous. Asynchrony means that the algorithm does not require low-latency communication. In fact, because dynamic programming planning is designed to handle stochasticity, robots can make a number of moves with a “stale” navigation function. Non-optimal moves can simply be viewed as stochastic actions. This property is well-adapted to the characteristics of real wireless networks.

Another interesting observation is that system performance could actually improve with random message failure. In the connectivity algorithm, nearby robots essentially compete for the chance to move, with the winner chosen arbitrarily. Random communication delays can give one robot an advantage over its neighbors, which acts like a fail-fast. The “losing” robots end their search earlier than they otherwise would have with perfect communication.

The experiments in this paper validate only the most basic capabilities of our approach. We plan to evaluate the system among various arrangements of obstacles, with larger teams of robots, and in splitting or merging operations. Important questions for future work include fully integrating formation reconfiguration with formation control, implementing cooperative localization that does not rely on active beacons, and consensus for determining new goal formations.

ACKNOWLEDGMENT

This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales (NSW) State Government.

REFERENCES