Multi-Robot Caravanning

Rachel Glockenmeier\textsuperscript{1}, Cameron Smith\textsuperscript{2}, Jory Denny\textsuperscript{3}, Andrew Giese\textsuperscript{1}, Nancy M. Amato\textsuperscript{1}

\texttt{r.glock@tamu.edu, noremac.knj@yahoo.com, jorydenny@cse.tamu.edu, awg4619@cse.tamu.edu, amato@cse.tamu.edu}

August 3, 2012

Abstract

In today's society, robotics is emerging as a common solution for many tasks, e.g. group coordination. In this paper, we present multi-robot caravanning as an application for distributed group coordination of movement amongst heterogeneous agents. One example is a group of agents traversing an area, both indoors and outdoors, when the agents only know a portion of the environment. Caravanning can be used in systems with many robots or in large environments where sharing a complete map of the area would be impractical, expensive, and in some cases, impossible. Caravanning is the problem in which a flock of agents, each having different representations of the environment, must elect a member to lead the flock to a known waypoint. For our project, we use the hallways of an office building to demonstrate a leader accurately traversing a large environment.

1 Introduction

There are many tasks that require groups of different types of robots working in conjunction with one another. Be it to play a game or traverse an area, it is often necessary for agents to communicate and work with other agents in their group or around them.

In this paper, we discuss the idea of multi-robot caravanning. For example, there may be a group of two robots traveling together in an environment with indoor and outdoor areas. Each robot has a distinct set of information about the area and capabilities. In this example, the first robot, $R_1$, can localize based on markers placed in the environment and the second robot, $R_2$, can determine its location based on GPS. In the indoor portion of the environment, markers are abundant and GPS is difficult to obtain, so $R_1$ will be better suited to lead the caravan. In the outdoor area, $R_2$ will be better suited to lead the caravan as the GPS will be available and there are fewer markers. Working together, the robots are able to safely traverse the entire area.

From these examples, it is clear that this idea has applications in search-and-rescue and military scenarios. One example from search-and-rescue is where one robot may be able to lift rubble, but another robot is able to locate the individuals trapped in the area. In this instance, both robots are needed to solve the problem, therefore, they must cooperate with one another. This problem not only applies to the specified domains, but in any situation where a group of different agents must travel through an environment to achieve goals.

We formally define the multi-robot caravanning problem as described in Definition 1.

\textbf{Definition 1.} Given a heterogeneous group of $n$ agents $R = \langle r_1, r_2, \ldots, r_n \rangle$ needing to cooperatively travel through a known environment $E$ along $m$ waypoints $W = \langle w_1, w_2, \ldots, w_m \rangle$.

Collectively, the group of agents knows the entire environment, however individual agents have various sensors, sizes, or representations of the environment $e_i \subseteq E$, where $e_i$ is the $i$th robot's representation/capabilities, and $\forall r_i, r_j \in R | r_i \neq r_j \land e_i \neq e_j$.

The multi-robot caravanning problem is thus defined as: Generate a valid path $P$ visiting each $w_k \in W$, assigning a robot $r_i \in R$ as the leader from $w_{k-1}$ to $w_k$. $w_0$ is the initial position of the group.
In this work, we detail our experimental setup and our progress toward addressing the problem of multi-robot caravanning through a leader election algorithm, which reduces the amount of necessary communication between agents. A leader is selected at each waypoint, and the rest merely enact a simple following behavior. In addition, our solution utilizes a roadmap-based approach in order to generate the path between waypoints [7].

The remainder of the paper is organized by: Section 2, which outlines some work related to our project, Section 3 explains the method we propose, Section 4 explains our experimental setup, and Section 5 discusses our results and future work.

2 Related Work

Our proposed solution requires the contributions of three main works. These three related works are: motion planning (Section 2.1), leader election (Section 2.2), and localization (Section 2.3).

2.1 Motion Planning

The goal of motion planning is to find a valid path given a start and goal configuration. There are several methods that can be used to determine such a path. In our research, we apply Probabilistic Roadmaps (PRMs) [7], which is a method of representing the connectivity of an environment. These roadmaps have been used in many applications, including those in pursuit-evasion [11].

The nodes of the roadmap are created by randomly sampling the space, after which the edges are generated as valid paths between the nodes. This map can then be used to construct paths through the environment from the given start to the goal configurations.

Our method uses different types of sampling methods to create different roadmaps. There are several methods documented in literature. These methods include Obstacle-Based PRM (OBPRM) [1] which generates nodes close to obstacles and Medial-Axis PRM (MAPRM) [13, 8] which generates nodes with maximum clearance from obstacles.

2.2 Leader Election

Leader election is the selection of a leader from a group of agents. In our approach, which is derived from the Bully Algorithm [3], at each waypoint a leader election takes place. In this leader election, each robot makes a bid to be the leader, this bid is based on how capable the robot perceives it is to lead the group. In many cases this bid is based on a path metric to the next waypoint. The robot with the highest bid is named the leader and the rest follow. This robot will remain the leader until the group reaches the next waypoint and another leader election occurs. This algorithm has a complexity of $O(n)$ where $n$ is the number of agents in the system sending bids.

2.3 Localization

Localization is the process of determining an agent’s position and orientation in an environment. In many cases, GPS is used to localize [5]. While this is a feasible approach outside, indoors it is unreliable. In our research, we only focused on indoor environments, so we applied marker-based localization.

Marker-based localization uses landmarks, or markers, with a known location and size. When the agent identifies a marker, it is able to calculate its position and orientation from a transformation matrix [6]. This information can be compared to a desired configuration. By doing this, the robot is able to accurately determine how it needs to adjust its plan to reach the next waypoint.

3 Methodology

Algorithm 1 describes the actions that each agent executes. First, a leader election occurs, which is detailed in Algorithm 2. If the agent is elected as the leader, it will extract a path from the roadmap to get to the next waypoint. The leader then tries to follow its roadmap to the waypoint, localizing when it detects markers and then continuing to follow its path. If an agent is a follower, it will perform some form of flocking, e.g., Boids flocking [10] where it modifies its trajectory to follow the flock without colliding. When the leader arrives at a waypoint, it will call for another leader election and the process will be repeated until the final waypoint has been reached.

In Algorithm 2, each agent determines its path metric (in this case, the distance to the waypoint) and makes a bid for the leader role based on its path metric. When all agents have bid, the agent that has the highest bid is made the leader. In this case, the agent with the highest bid is the one with the shortest path. This process occurs at every waypoint until the flock makes their way to the final destination.

When Algorithm 1 is used, it is assumed that the agent already has a roadmap, as the creation of the roadmap is not part of our algorithm. The creation of the roadmap itself is done using the sampling methods described in Section 2.1. However, in this paper we only consider the use of two agents.
Algorithm 1 Agent Algorithm Overview

Input: Waypoints \( W = \{w_1, w_2, \ldots, w_m\} \)

1. for all \( w_k \in W \) do
   2. \( \text{leader} = \text{ElectLeader()} \)
   3. if \( \text{leader} \) then
      4. \( p = \text{roadmap}.\text{FindPath}(w_{k-1}, w_k) \)
      5. Traverse \( p \) while localizing
      6. Call for leader election
   7. else
      8. repeat
      9. Flock with group
   10. Until Leader election call
  11. end if
  12. end for

Algorithm 2 ElectLeader

1. Broadcast ID and path metric
2. Receive \( M \) as a map of IDs to path metrics
3. \( \text{bestID} = \arg \max_{i \in M} M[i] \)
4. if \( \text{bestID} \equiv \text{myID} \) then
   5. Broadcast end of leader election
   6. return leader
5. else
   7. return follower
9. end if

4 Experiments

4.1 Experimental Setup

For our experiment, we used iRobot Creates, disk-shaped robots, 330mm (13 inches) in diameter and 114mm tall (4 1/2 inches). The Creates have a forward-facing bump sensor that can be used in reactive obstacle avoidance. We equipped our Creates with an Asus Eee netbook running Ubuntu, which has network communication abilities and is small and light enough to fit on top of a Create without hindering its movement. Eees also have built-in webcams which are used to visualize the environment and view wall markers. We used Player [4] to interreact between the netbook and the robot. To create an actual model of the environment, we used Maya modeling software. This model is used for sampling the environment to create a probabilistic roadmap. Our roadmap was generated using the Parasol Motion Planning Library (PMPL) that was developed at Texas A&M University. In the experiments, we used Uniform Random Sampling and Grid Based Sampling to obtain a roadmap of the environment.

In order to prepare our environment for robot traversal, we positioned more than 60 markers throughout the hallways of the 4th floor of the H. R. Bright Building on the campus of Texas A&M University. Created using ArUco [9], these markers are used for robot localization during navigation. We use OpenCV [2] to capture images, this allows ArUco to interpret images from the webcam. Each marker was placed strategically based on the complexity of different areas of the environment – namely those with obstacles (trashcans, tables, benches, etc.). We chose a marker size that could accurately be interpreted by the robot from a relatively far distance (approximately 3 meters).

4.2 Results

We have tested our leader algorithm on a single robot, attempting to have the robot navigate through the environment (hallways of the Bright Building). Successful navigation was based the accuracy of the robot’s actual odometry, based on the path generated from an initial waypoint to a predetermined goal configuration. Evaluating the time and accuracy of completion, we ran a series of trials to test the accuracy of the robot’s path navigation. On average, we found that the robot was able to traverse its path metric — a distance of over 10.5 meters (34.5 feet) — in 5 minutes 26 seconds, with a standard deviation of roughly 34.2 seconds and at most 3 collisions. In Figures 1 and 2 below, white space represents the traversable area while black, red and green objects represent obstacles.

![Figure 1: Given the start and goal configurations, this is the path that was extracted from the roadmap.](image)

Because Creates have imperfect odometry, having the robot accurately traverse the environment is chal-
lenging. However, Figures 1 and 2 show a close similarity between the real path and the actual path generated based on odometry. Currently, our leader and follower algorithms are naïve, in that we exclusively assign each robot as either the leader or a follower. Based on its localization and odometry, the robot was able to successfully maneuver from the start to end configuration.

5 Conclusions and Discussion

Multi-robot caravanning is an application that can be applied in a variety of situations, particularly those requiring coordinated movement among a group of heterogeneous agents. With multi-robot caravanning, this is done using minimum communication among agents, through the election of a leader. We hope to expand this work to include multiple robots in the near future. We anticipate creating a more robust follower algorithm, which currently is only told to remain a certain distance from the leader. Additionally, we would like to incorporate an algorithm that enables waypoints to be generated dynamically based on the distance from the previous waypoint. Also, we would like to incorporate our market-based leader election to allow a leader to be chosen based on shortest distance to the next waypoint. Furthermore, we would like to add a Kalman filter [12] to account for sensor noise.

References


