Experiment Design for Large Multi-Robot Systems

James McLurkin
Computer Science and Engineering
University of Washington
Seattle, WA, 98195
jamesm@cs.washington.edu

Abstract—Multi-robot systems present unique challenges for the experimenter. There are three key components for conducting successful multi-robot experiments: 1) Defining platform-independent metrics for evaluating algorithm performance, 2) determining the ground truth of the robot's physical configuration and network connectivity, and 3) designing a usable system, with centralized data collection and hands-free operation. These three components require specific hardware and software unique to multi-robot systems. This paper presents a brief survey of the literature on the key challenges facing experiment design for multi-robot systems, and presents the author’s experiences with working with dozens of robots on a daily basis.

I. INTRODUCTION

Experiment design for large multi-robot systems presents unique challenges for the researcher. In this paper, we discuss three of these challenges: defining performance metrics that capture the algorithmic accuracy of a group of robots, designing custom hardware for experimentation, and building a user interface to make working with a swarm of robots practical. We survey the current state-of-the-art in multi-robot experimentation, briefly discusses our experimental setup, and conclude with some key challenges for future systems.

II. PERFORMANCE METRICS

Performance metrics for multi-robot systems measure how well the group of robots performs a specified task. There are many things that affect the performance of a multi-robot system: algorithmic correctness, robot execution, and network performance. The best performance metrics are those that produce platform-independent results, suitable for comparisons across multiple systems. Ideally, performance metrics are dimensionless, and not directly connected with any particular robot dimension or population size.

One standard approach is to measure the performance of a multi-robot system by comparing the observed performance with the best and worst theoretical performance. For example, the best performance possible for a navigation algorithm is to move the robots along the shortest navigable path from the start to the destination at the fastest possible speed. This leads to a natural accuracy metric: the path efficiency, which is defined as the ratio of the best path to the observed path[19]. This metric produces results between [0, 1] and can be used with any type of hardware. A less obvious example is the accuracy of an algorithm to arrange the robots into a hexagonal lattice [28]. In an ideal configuration, the angles between edges adjacent to each robot should be spaced at angular intervals of \(\frac{\pi}{3}\). The lattice quality is computed by measuring the average error from these discrete ideal angular arrangements over the entire group of robots.

One of the difficulties with defining these metrics lies in describing the theoretically best configuration and the worst possible configuration. Often, it can be unclear what the worst possible configuration is. One possible choice is the antagonistic configuration, such as robots that disperse when clustering is the goal. However, the antagonistic configuration is usually very unlikely, so sometimes a more appropriate lower performance bound is a random configuration, such as robots that take on a uniformly random distribution of tasks when a fixed distribution of tasks is the goal[18]. Normalizing the accuracy metric to one or the other of these worst-case configurations will produce different results, and it is up to the algorithm designer to select the most appropriate accuracy metric for the algorithm and its application.

Multi-robot systems have two additional quantities that all algorithms should be measured against: the population size and the rate of change of the communication network. Scalability with respect to population size and robustness to robot failures is critically important to multi-robot systems. The scaling of communications usage can often be calculated analytically [13], but robustness must be measured empirically. Experiments should be designed to test key algorithm properties: convergence time, communications usage, and accuracy, against as many robots as can be fielded. Many algorithms that scale well in theory break down on large populations of actual robots. For example, several researchers have shown that interference in multi-robot systems can reduce or eliminate efficiency gains by increasing population size[10], [14].

It is clear that the mobility of the robots in a multi-robot system can have dramatic effects on the communication network, but it is unclear what is the best metric for quantifying the rate of change of network topology. Potential metrics include network half-life, which is the interval of time before half of the neighbors of a robot have changed [15]. The half-life time is \(\min((ln2)/\mu, n/\lambda)\), where \(\mu\) is an agent's rate of departure, \(\lambda\) is the rate of arrival, and \(n\) is the number of robots. Another measure is to examine the matrix of neighbor transition probabilities. In a fixed topology, this matrix will have ones in the cells corresponding to the network connectivity between robot \(i, j\), and zeros in all other entries of the matrix. In a well-mixed scenario, this entire matrix will approach a uniform distribution.
Although it is unclear how to best measure network topology changes, it is clear that there is a practical upper limit on the rate of these changes. At this upper limit, the robot's ability to reconfigure the network will outstrip the network’s ability to keep neighbors and routing information up-to-date. This ultimately places an upper bound on robot mobility, limited by communications bandwidth. The robot speed ratio\(^1\)\(^7\), is the ratio of robot speed to message speed, and is a straightforward way to estimate this upper limit, and design systems that operate well below it.

III. HARDWARE DESIGN FOR EXPERIMENTATION

Experimentation on multi-robot systems often require custom hardware and software that is specific for data collection, but is not used in algorithm execution. There are three main hardware features that the hardware must provide. First, there must be a way to measure the ground truth of the robot’s physical configuration and network connectivity. Second, the individual robots must be able to measure some features of their physical configuration and network connectivity. Finally, programming and debugging should be as hands-free as possible, as physically handling many robots for these common tasks will reduce user efficiency. We address hands-free design in the section on usability and consider hardware for geometry measurements in this section.

A. Ground Truth Geometry Measurements

Multi-robot experiments require a data collection system capable of simultaneously measuring the ground truth of the robot’s configuration and the communication network. Measuring network connectivity is a straightforward task, but does require either multi-robot logging software to collect realtime network connectivity data, or synchronized logs from multiple robots. Measuring ground truth positions of the entire system of robots is more difficult. There are three common types of systems for determining geometric ground truth in multi-robot systems: GPS receivers, centralized vision-based systems, and radio-acoustic ranging systems.

Outdoor robotic systems can use GPS receivers to determine position ground truth. The Ratler multi-robot system from Sandia National Labs \(^8\) uses GPS receivers for localization. However, many application environments and most laboratory environments are indoors, where GPS signals are not readily available, so we focus our attention on indoor environments.

A few multi-robot systems use radio-acoustic ranging systems. The system used by Mataric et al. \(^6\) uses two base stations to triangulate the global position of each robot, but cannot determine the global heading directly. The cricket system \(^26\) uses transceivers mounted on each robot to measure inter-robot distances, which can be combined to compute global robot poses. Distance measurements from these systems have high accuracy, but computing the global pose of individual robots from local measurements can be challenging \(^25\), \(^6\), \(^8\).

Camera-based tracking systems are currently the most common method determining ground truth of the robots’ configuration in an indoor environment. These systems must have the ability to uniquely identify individual robots in the camera image. One approach is to use a 2-D bar code on the top of each robot, which can be used to uniquely identify individual robots and determine their global pose, \(\{x, y, \theta\}\). The bar code requires a significant amount of area on the top surface of the robot, which can make it difficult to scale to smaller robots. The use of bar codes also makes it difficult to place status indicator LEDs on the top of the robots, which is a primary user interface in multi-robot systems. Another camera-based approach is to use IR emitters on the top of each robot to transmit a unique message to a centralized camera. If multiple emitters are used on each robot, the global pose of each robot can be determined. However there is a minimum separation distance required between multiple emitters so that the camera can distinguish them and determine a robot’s heading. For small robots, a single emitter may be the only practical option, which allows only direct measurement of position, \(\{x, y\}\), but not global heading. However, if the robots are able to measure their local network geometry, then global heading can be estimated by combining local estimates of bearing to neighbors and the global measurements of neighbor positions.

B. Local Geometry Measurements

Estimating the local network geometry, i.e. the edges and angles of the geometric graph in which the robots are embedded, is usually a requirement for multi-robot configuration control. There are two approaches to solving this problem. The first approach is for each robot to estimate its global position in an external coordinate frame, then combine this with network connectivity measurements to produce local network geometry. The second is for each robot to measure the relative positions of its neighbors directly. In this approach, each robot uses a custom sensor array to measure the poses of its neighbors relative to its own coordinate frame.

In some multi-robot systems, each individual robot can estimate its position in a global coordinate system. In outdoor environments, access to GPS information is available. In indoor environments, the position measurements from the ground-truth tracking system can be communicated to the robots. While this is a pragmatic solution, it does not accurately capture the kinds of geometric information available to robots in uninstrumented environments, where the robots must use their local sensors and measurements to estimate their global position. A better approach is that of robocup soccer, where the AIBO robots use visual landmarks to determine their global coordinates. Robots large enough to carry laser scanners can determine their global location based on local measurements \(^5\).

If the infrastructure for a global coordinate system is not available, the local network geometry can be measured between each robot and its nearby neighbors. Multi-robot systems need a specialized sensor to measure the local network geometry. There are many approaches to measure this information, including vision-based systems \(^12\), \(^11\), and infrared light \(^19\), \(^27\). The local network geometry is a
good model of what an individual robot can measure locally. It also imposes important algorithmic constraints on the system, such as the inability to compute an exact Voronoi cell without communicating with robots further away than neighboring robots [3], [1].

IV. USABILITY: THE USER INTERFACE

Good experimental methodology also includes building a usable robot system. This is especially critical in multi-robot systems, where the details of interacting with any single robot scaled by the total population can occupy a disproportionate amount of the researcher’s time. The easier it is to modify software, run experiments, and collect data, the more this development cycle can be repeated. The only way to achieve this is to design usability features into the system. A successful design philosophy is to focus on hands-free operation and produce a “hardware simulation”: a system that combines the benefits of working with real hardware with the ease of use of a simulation.

The key physical components of a usable multi-robot system are centralized command and control, remote programming, intuitive user interface [23], [7], and data logging integrated with ground truth measurements. Some researchers have systems with these properties [20], [24].

One of the most difficult validations of good system design is the classroom laboratory environment [22], [2]. Time constraints, limited resources, and the pragmatic approaches of students to finish the assignment can strain even the best system design and experimental guidelines. Any classroom system must be highly usable, as students must get past the learning curve quickly to complete assignments in a timely fashion.

V. A CASE STUDY: SWARMBOT EXPERIMENTAL SETUP

The SwarmBot robot platform [19] shown in Figure 1 is designed to facilitate development and experimentation on multi-robot systems. This section describes the key features of the hardware that facilitate experiment design and operation. We then describe the experimental protocol and show data from a representative experiment.

A. Data Collection Infrastructure

The SwarmBots are autonomous, using only local computation and sensor readings to run their algorithms. Each robot has a 32-bit ARM processor running at 40mhz, a unique ID, and bump sensors. Three large, top-mounted LEDs are used to display information about the robot’s internal state. The robots were designed to be used as hands-free operation as possible. Software updating and data collection are done wirelessly, and the robots can autonomously dock and charge themselves. Figure 2 shows several robots docking and charging.

Fig. 1. a. Each SwarmBot has an infra-red communication and localization system which enables neighboring robots to communicate and determine their pose relative to each other. The three lights on top are the main user interface, and let researchers determine the state of the robot from a distance. The radio is used for data collection and software downloads. b. There are 112 total robots in the Swarm, but a typical experiment uses only 30-50 at a time.

Each robot has an infra-red communication and localization system that allows nearby robots to communicate with each other and determine the pose \( p = \{x, y, \theta\} \) of their neighbors [19]. We usually run the system at its lowest transmit power setting, giving each robot a communications range of about one meter. This produces multi-hop networks within the confines of our lab.

A diagram of the complete data collection system is shown in Figure 3. There are four main components: the robots running the experiment, a single host robot, a ceiling-mounted infrared(IR) localization camera, and a computer with data logging software.

Ground truth positions of the robots are determined by a vision-based localization system. The system uses a single IR emitter on the top circuit board of each robot, and tracks the \( \{x, y\} \) positions of all of the robots simultaneously. This emitter transmits an encoded pattern that allows the vision system to decode 10 bits of data per second per robot. Each robot uses its emitter to transmit its robot ID, which allows us to identify its individual position in the recorded data. Figure 5 shows the calibration data and position error histograms. The system was calibrated by manually measuring positions of 12-20 robots in the environment, then using linear regression to produce a conversion from pixels to millimeters. The linearity is quite good, and the mean error is 15.4 mm. The system reports the positions of all the robots once per second. Because of the 1 Hz update rate, we limit the maximum speed of the
Fig. 3. Diagram of the SwarmBot data collection system. Each robot has a single top-mounted infrared emitter that flashes with an individual pattern. The ceiling-mounted camera identifies each robot with this pattern and tracks the positions \( \{x, y\} \) of each robot simultaneously, reporting the results to the computer at 1 Hz. The host robot uses its radio to query each experiment robot and reports their logged variables to the computer. The computer unifies both data streams to present a real-time graphical display to the user and log data for future analysis.

robots in all experiments to 80 mm/s.

A single host robot uses a radio network to query all of the other robots for their telemetry during execution. With 25 robots, the frame rate of this system is around 1 Hz. This data is sent to the data logger program shown in Figure 4a. This program integrates the camera positioning data with the telemetry to produce a composite log. During the process, the logger also combines the local neighbor pose estimates with the global positions to produce an estimate for each robot’s global heading.

The system is effective, but the limited field of view of the single camera and the ceiling height limit the workspace to about 3 m \( \times \) 4 m. Also, the limited memory available on the robots prevents logs from being stored locally, so the rate of data collection is limited by the radio bandwidth and the number of robots in the experiment. This limits the temporal precision of the collected data.

B. An Example Experiment

This section describes the setup and protocol for a typical experiment with the SwarmBots. Our recent work is focused on measuring algorithm accuracy in dynamic network topologies [17]. We designed an experimental protocol to produce results that would allow us to compare the performance of ten different algorithms [20] at a variety of speeds. The algorithms are for a variety of applications, ranging from navigation [4], dynamic task assignment [18], and boundary detection [21]. Each algorithm is run on 25-35 robots moving randomly around a bounded workspace of 2.43 m \( \times \) 2.43 m (8’ \( \times \) 8’) square. Because their communication range is only about one meter, the constant motion generates many changes in the network topology.

The movement is controlled by the BUMP\_REFLECT motion behavior. This behavior moves each robot in a straight line until it contacts an obstacle, then uses the bump sensors to estimate the angle of incidence and “reflect” the robot back into the environment. Figure 6a shows the path of a single robot running this behavior, and Figure 6b shows the combined paths of many robots. The behavior does a good job of keeping robots dispersed uniformly throughout the environment and away from the walls.

Each experiment tests the algorithm over a wide range of robot speeds and communications parameters. The robot speed and communication parameters are combined to calculate the robot speed ratio (RSR) [17]. The RSR is the ratio of robot speed to message propagation speed which produces a dimensionless metric. A robot moving at a RSR greater than one away from the source of a message will never receive this message, and is essentially disconnected from the network, so we use \( \text{RSR} = 1 \) as our maximum speed. We tested each algorithm in a static configuration and over a range of robot speed ratios from 0.005, to 0.640. This range of

Fig. 4. Views from the overhead camera and data logging software. There are two overhead cameras, one for recording video, and the other for logging the positions of the robots. The data logging software displays selected state for each robot the robots and the communication network. In this image, a sub-graph of the global boundary is displayed in the viewer. Visualizing the communications network and robot state in real-time is a valuable debugging aid for distributed algorithms.
The calibration data and error histograms for the vision-based localization system. The x-axis shows measured error and the y-axis shows the bin population. The mean error over x and y axes is 15.4 mm. The system occasionally produces errors in excess of 30 mm, but refused to do so when we were testing it. These large errors are outliers in the error distribution, and are easily filtered out with simple heuristics.

Example experiment results are shown in Figure 7. The algorithm under evaluation was a distributed boundary detection algorithm [21]. Figure 4 shows the boundary detection algorithm executing on the robots. Robots with their blue lights on, and those colored blue in the data logger screen capture have classified themselves as boundary robots. Those with red lights have classified themselves as interior robots. Those with blue lights have classified themselves as boundary robots. The robots rotate twice the measured angle of incidence, “reflecting” off of the obstacle and back into the interior of the workspace. The behavior is effective at causing robots to change neighbors frequently, and keeping the density of robots roughly uniform throughout the environment.

All of the algorithms were tested while the robots were executing the same motion behavior in order to test the robustness of the algorithms to rapidly changing network configurations. The BUMP/REFLECT behavior drives the robots in a straight line until they contact an obstacle. The robots rotate twice the measured angle of incidence, “reflecting” off of the obstacle and back into the interior of the workspace. The behavior is effective at causing robots to change neighbors frequently, and keeping the density of robots roughly uniform throughout the environment.

The samples for computing statistics. Future work on metrics should consider this problem more carefully.

VI. CONCLUSION

Multi-robot systems require specialized performance metrics, hardware, and user interface design for successful experiments. Because of the realities of working with large numbers of robots, the usability of the system must be carefully considered. Two different measurements of geometry are often required, one to determine ground truth of the configuration...
and network connectivity, and another that the robots use to measure the local network geometry. Experimental protocols need to be designed to measure sensitivity to network and population changes to understand the limits on real-world performance. With careful design and cross-platform validation, experiment design for multi-robot systems can allow algorithms and techniques to be applied broadly throughout the field. This will advance both the theoretical and practical understanding of multi-robot systems.

ACKNOWLEDGMENTS

The author would like to thank his University of Washington CSE491 class for many insights on applications for the classroom and Fayette Shaw for discussions on measuring topology changes in multi-robot systems.

REFERENCES