Multi-Robot Long-Term Open Polyline Patrol

Sara E. Lahr
Division of Science and Mathematics
U. of Minnesota, Morris; USA
lahrx019@morris.umn.edu

ABSTRACT

Multi-robot patrol is an ever growing field in which a team of robots works to optimize the frequency in which they pass through certain points. The number of applications for this is large and includes garbage collection, cleaning, and surveillance. Previous work focused on the uniformity of the patrolling algorithm. We present an algorithm that works towards high-frequency patrol but focuses on long-term maintainability. A team of robots patrols a region along a fence. Each robot patrols its own segment of the line and removes itself to return to the charging station when it runs low on power. A new robot is then released, and the robots adjust the organization of the team accordingly. This algorithm presents a way to maintain coverage of an area over a long period of time as the robots are able to recharge and then return to patrolling.

Keywords

Multi-robot, Scribbler, patrol, long-term

1. INTRODUCTION

Patrolling is the task of covering a given area repeatedly and is well suited for a team of robots. The purpose of patrolling ranges from garbage collection, cleaning, and surveillance. A common goal of patrolling applications is high-frequency coverage; we want the maximum amount of information about each point in an area as possible. Different aspects of this goal are uniformity of visits, maximal average frequency, and maximal minimum frequency. These can be used to judge the coverage of both an area [5] and a line [4]. Other algorithms have a different goal and instead try to provide protection against adversaries [2]. Although these algorithms work well in accomplishing short term goals, they do not always consider the limitations of robots. Modern robots do require recharging, a detail that is often overlooked. Our algorithm takes this into account, and our goal is the longevity of the system from one application to another.

The area different algorithms focus on covering varies greatly. Much existing work focuses on coverage within a two-dimensional area. This type of algorithm can be generalized to the coverage of a circular region or a closed polygon. An algorithm that creates a path through an area can be considered as simply creating an one-dimensional path to be covered. With a circular path, any point along the region is accessible by any other point from either direction. The circular nature of the path makes uniformity simple as the robots can all travel in the same direction. This work focuses on patrolling along an open-polyline, a line with an end on each side which makes it more difficult to provide frequency guarantees.

The area is split into equal length segments, and a robot is assigned to patrol each one. There exists a central tower or holding area for all inactive robots. When a robot runs low on battery, it removes itself from the line and makes its way back to the tower. The other active robots are informed of the now empty segment. All robots between the empty segment and the tower move over a segment until the empty position is next to the tower. A new robot is then released from the tower, and the algorithm continues as before until another robot needs to be reinforced.

To evaluate the algorithm, we created our own small "fence" within our lab. We patrolled it using Parallax Scribbler robots. The robots were preprogrammed and centrally controlled through a Python module. The remainder of this paper is structured as follows. Section 2 reviews related works. In Section 3, we discuss the long-term patrolling algorithm. The hardware and experiment details are explained in Section 4, and Section 5 presents a few preliminary results. We discuss the results in Section 6 and then conclude with Section 7.

2. RELATED WORK

Coverage of a region by a team of multiple robots can be approached in a number of different ways. Coverage of an area is a very common task. Choset summarizes recent findings and explains different techniques[3]. The biggest distinction between different algorithms is the way that they break up the space. This is a measurement of how the barriers between cells (or regions) covered by a single robot are defined. Approximate cellular decomposition breaks the space into equally sized cells, but the union of the cells is only an approximate description of the entire area. Semi-approximate decomposition slices the space into cells of equal width with the top and bottom expanding to the shape of the space. Our algorithm most closely resembles the third technique, in which the area is decomposed into identical cells that cover the entire length of the fence.

Once the area's cellular division has been determined, coverage algorithms require the creation of a path that covers the area. Some are more flexible and create paths based on the amount of communication available[6]. With unlimited communication, the paths are not calculated beforehand; instead the robots act on the information received from their team. Unfortunately, the robots used in our work are unable to communicate as efficiently and are unable to be as adaptive within an open environment. For this reason, we took an approach more similar to [1]. Here, the authors still use

some communication to adapt to situations where a robot failed, but a static path is initially created for the robots. In the same way, our algorithm assigns a static segment for each robot to cover, but once a segment needs to be reinforced, the robots are dynamically reallocated.

Coverage of an area can easily be extended to the patrol of an area. However, where coverage seeks to visit every point only once, patrol requires every point to be visited as often as possible. The authors of [5] focus on achieving optimal frequency by generating a path that covers all space in the area. The robots are then uniformly distributed and patrol along this path. While their goal of high-frequency patrol is similar to ours, their path through the space is circular. Because there are no ends to the path, uniform frequency is much easier to achieve.

Patrolling a line is similar to patrolling an area except the path taken now resembles an open polygon or an open-ended fence. This requires a more advanced patrolling algorithm. While the path is easier to generate, it is much harder to patrol uniformly. One technique used is to create an overlap of the segments such that the robots move in and out of areas covered by their neighboring teammates[4]. This helps to accomplish consistent coverage. While our work is similar in that we break the line into segments for each robot, it is different because we focus less on synchronized, uniform patrol and more on long-term patrol.

Little work has been done in patrolling a region over the long-term. While high-frequency coverage is important, it cannot be maintained if the robots are unable to recharge and be redeployed. For this reason, our algorithm focuses on the flexibility of patrol such that when a robot is running low on power, it can remove itself from the line, and the other robots will adapt to cover the now empty segment.

3. OPEN POLYLINE PATROL

3.1 Assumptions

We made very few assumptions to complete our algorithm. The robots are not synchronized, and we do not assume that they move at equal speeds. In addition, the robots cannot communicate with each other; they can only communicate with the central computer.

For our experiments, we do assume that there is a human being present at the central tower. Once a robot returns, the person changes its batteries and places it into the ready queue. At some point this could be automated, but this was not part of our goal in this project.

Currently, we need enough robots to cover all the segments. If too many robots fail such that there are not enough robots to cover all of the segments, the functioning robots will continue to patrol segments, but there will also be some unpatrolled segments.

3.2 Algorithm

Even though our algorithm may not be completely robust, it completes the task of long-term coverage given enough robots are available. This is hardly surprising, however, because little can be done in the way of coverage once all the robots begin to fail.

Our algorithm begins with all available robots stationed at the tower. See Alg. 1. Lines 3-7 initialize the positions of the robots to their respective segments. For every segment that needs to be filled, a robot is removed from the waiting queue within the tower and becomes active. A thread is started; each thread represents the individual robot's actions (Alg. 2). By the end of line 7, there is a robot patrolling every segment. They will continue to patrol their segment until they either run out of battery or are needed to patrol a different segment left vacant by another robot. Lines 8-19 of Alg. 1 depict the loop that continues until no more robots are functioning. If at any point a robot needs to leave its line, it will report that its former segment is now empty. The central control will then send reinforcements. All robots which are patrolling segments in between the empty segment and the central tower stop patrolling, move down a segment, and start patrolling a new segment. After this slide, the empty segment will be the first segment from the tower. The tower releases a new robot in the same way that the robots were released at the beginning of the algorithm, and all segments are again covered.

Algorithm 1 Patrol Main Loop

Patrol

```
1:  Initialize robots
2:  Initialize robots' segment numbers to that of tower
3:  for all seg in line do
4:      startThread(bot, seg)
5:      bot.STATUS = ACTIVE
6:      goToSeg(bot, seg)
7:  end for
8:  repeat
9:      if A segment is not being patrolled then
10:         for all bot where bot.STAUS = ACTIVE do
11:             if bot is patrolling a segment between empty segment
                and tower then
12:                 Move bot over one segment
13:                 Adjust segment number of bot
14:             end if
15:         end for
16:         startThread(bot, empty_segment)
17:         bot.STATUS = ACTIVE
18:     end if
19: until All robots have failed
```

Algorithm 2 Actions of Individual Robots

startThread(*bot*, *seg*1**)**

```
1: bot moves to seg1
2: repeat
3:     bot patrols line
4:     if Another segment is empty and bot needs to move to seg2
       then
5:         killThread
6:         startThread(bot, seg2)
7:     end if
8: until bot runs out of battery or is needed elsewhere
```

4. EXPERIMENTS

4.1 Hardware

For this work, we employed Parallax Scribbler robots. Marketed as toys, the Scribblers (See Fig. 1) have limited sensor capabilities. They come preconfigured and can do a number of tasks without any programming on the user's part. We used them because they are relatively inexpensive and available in larger quantities.

Scribblers come equipped with a BASIC Stamp®2 microcontroller that is easily accessed by a computer through a serial port. There are three light sensors and two infrared sensors on its front side. Underneath the Scribbler, there are two line sensors that use infrared to distinguish between light and dark and return a binary response. Basic output is achieved through a speaker and three LED lights. There are two wheels controlled by independent DC motors along with a stall sensor to respond when the Scribbler becomes stuck. Scribblers come with software for accessing and programming the firmware. The Scribbler can be programmed both

Figure 1: Scribbler robots with Flukes.

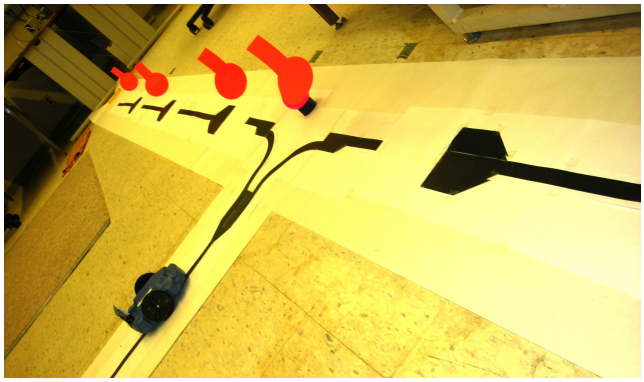

Figure 2: Part of the experimental setup. A lone Scribbler leaves the tower and heads towards the line. It will turn either left or right onto the line before it starts jumping to other segments. The hanging orange targets seen are used to guide the Scribbler when it needs to follow the line backwards.

via a graphical interface and text (PBASIC).

The Scribblers also come with the Fluke, created by Georgia Tech. It contains a Python module that allows for higher-level access to the Scribbler. The module was created by the Institute for Personal Robots in Education (IPRE). The IPRE Fluke is Bluetooth compatible, allowing the Scribbler to be accessed wirelessly. The Fluke adds two LEDs, three infrared sensors, a battery voltage sensor, and a color camera to the available sensors.

Some aspects of our algorithm were implemented to work around the limitations of the Scribblers. To better understand the reasoning behind these decisions, some of the limitations are mentioned here. The Scribblers lack odometry, making exact, repeatable movements or turns nearly impossible. The light sensors and infrared sensors can be inconsistent as the values vary greatly depending on the light source or the texture of the ground surface. The Scribbler is not able to initiate any sort of communication, requiring the central control to query the Scribblers if any information is needed from the robot.

4.2 Architecture

To be able to control many Scribblers at once, we designed an architecture that was split into three layers. This allowed for the Scribblers to act autonomously when needed and upon the instructions or queries sent from the central control. The central control could not communicate with all the Scribblers at once, so this setup was necessary.

Layer 1

Layer 1 is the lowestlevel of control and consists of the firmware on both the Scribbler robot and on the connected Fluke. The Scribbler firmware, written in PBASIC, communicates directly with the Scribbler hardware; it controls the motors and is able to get direct sensor data. In addition, the ability to line-follow was added to this level. This function requires immediate feedback from the infrared line sensors underneath the robot. Adding this function at this layer allows the Scribbler to line-follow as a default behavior.

Attached to the Scribbler is the Fluke, programmed in C. The firmware on the Fluke handles all the image processing. Also, in the same way that line-following is a function on the Scribbler firmware, the ability to find the central tower was added as a function on the Fluke. When returning to the tower, the Fluke is constantly taking pictures and reacts immediately by heading in the appropriate direction.

Layer 2

This layer handles the communication between Layers 1 and 3. Written in Python, it is the base for all upper-level functions. For example, it takes a command in Python and translates it so the Scribbler or Fluke can interpret it. All basic functions, like moving and querying the Scribbler, are done at this level. All queries return information at this level. It also contains all the information for setting up the Bluetooth connection between the Scribblers and the computer.

Layer 3

Layer 3 contains the actual patrolling algorithm. These high-level functions use all the lower-level functions within Layer 2. This layer also controls all the threading. Each robot in action is controlled within a thread, and the thread terminates when the robot is finished. At all times, this layer is able to communicate with any robots connected over Bluetooth, whether they are patrolling the line or not.

4.3 Environment

To test our line patrol algorithm, we created a physical line for the Scribblers to follow (See Fig. 2). This line is cut into equal-length segments, with segments separated by white space. Each line segment is patrolled by one Scribbler. For our experiments, we created six line segments, three on either side of the central tower, which acts both as a recharging station and as the location of robots in a queue waiting to be deployed. Ten Scribbler robots were available to patrol.

We added a few features to the line to make up for the Scribblers' shortcomings. The lines are black, but every other part of the surface is white. Each line is thirty-four inches long with an eight inch approach region on one edge. The space between the segments is nine inches (the minimum amount of space to prevent Scribblers on adjacent segments from colliding). Over each line we added an orange target to help the Scribblers visually stay on the line. We added two brightly colored cones that the cameras on the Flukes could track. These were used to guide the Scribblers back to base and onto the line.

Although the Scribblers had more sensors available, we only used select sensors for added simplicity. The infrared line sensors were used to follow the line forward. Because the line sensors are not centered underneath the robot, we could not use them to follow the line backwards. Also, without accurate odometry, we could not turn the Scribblers around to go back along the line. Instead we used the camera to track the orange target hanging above each line. The camera was also used as a way to straighten the Scribbler on

the line as well as to track the marker that defined the tower.

4.4 Goals

Throughout the experiment, we kept track of a number of variables. We were most interested in the amount of time that each Scribbler spent idle in the queue versus the time it actively patrolled the line. Maximizing active time and minimizing idle time would allow us to achieve maximum efficiency. To determine the frequency of how often each line was patrolled, we also recorded the number of passes a Scribbler did on each segment as well as the total number of passes each Scribbler did on all the segments.

5. RESULTS

At this point, we were only able to run preliminary experiments and tests as the system is not yet completely operational. These tests do not return enough data for us to accurately analyze the system, but they do return enough figures to get an initial idea of the algorithm's performance.

We did a few runs to test that the robots were capable of relieving their teammates and returning to base. Table 1 shows the results from three of these runs. We were using three robots to patrol two segments, so at any given time, one robot was idle. Also, because we were testing the accuracy with which the Scribblers adjusted, we limited them to no more than two passes before they returned to base. This table gives an accurate idea of the timings associated with different actions. For these runs, the robots started on the segments they were to patrol. In the first run, both Anders and Caprica finished two passes on the line in roughly 26 seconds. From here, Leoben was released from the queue before starting to patrol. Its approach to the line accounts for the extra time. From that point on, the robots are released from the queue and accumulate the amount of time spent in different locations. There is a large inconsistency with this data as the Scribblers experience various physical problems with reaching the line and patrolling the segment, but with more robots and segments, these numbers will begin to balance out. The variance introduced by these physical difficulties will become trivial.

We then expanded the experiment to include two more segments and two more robots (See Table 2). This time we ran the robots until they ran low enough on battery to force them to return to base. We did not use fully charged batteries in order to keep the experiment to a reasonable amount of time. The robots were versatile enough to switch between the needed segments. Table 2 displays the switching of robots between segments. The segment number is the location of the segment within the line. Note that the segments are numbered 1, 2, 4, and 5 because segment 3 is the base. During the experiment, we also had a robot experience a communication error. After Hera returned and the batteries were changed, it was not able to reconnect. Nevertheless, the other robots were able to account for the loss of a teammate.

6. DISCUSSION

Even though we are still in the preliminary stages of collecting data, we are able to make a few generalizations based the behavior we have seen so far. First, we realize that the amount of idle time and active time depends greatly on the ratio of robots to segments. Too many robots in the queue results in more time spent idling. Also, as the number of segments grows, the amount power that is needed to reach the furthest segments also increases. The robots that return most often are the ones furthest away, as they have already expended power patrolling closer segments before being sent to reinforce the outer segments.

In addition, a very accurate battery voltage sensor is necessary. A few times throughout the runs, a sensor would return an impossibly low level. If this occurred, the robot would return to base even though it still had enough power to continue its task.

Lastly, the system is very dependent on communication between the central control and the Scribblers. Concurrent commands would limit the amount of time robots spend idle. There may be a task for the robot, but until the central control is able to instruct the robot on what that is, it will remain in its current state. Timings vary greatly depending on the current state of the system when instructions are needed.

Table 1: Initial timings using three robots to patrol two segments. The times are cumulative for each Scribbler. Horizontal lines separate individual experiment runs (3 total)

Name	Passes	Total Passes	Active Time	Idle Time
Anders	2	2	25.61749601	0
Caprica	2	2	27.65314102	0
Leoben	2	2	83.98272991	25.61783504
Anders	2	4	108.7633421	27.60852504
Caprica	2	4	97.66184402	79.18434095
Leoben	2	4	174.581063	61.24988103
Caprica	1	1	45.93947506	0
Anders	2	2	58.04915404	0
Leoben	2	2	56.32050991	71.99918509
Anders	2	4	134.5267649	25.76582193
Leoben	0	2	106.6047308	115.4285653
Caprica	2	3	80.16399407	105.5323648
Caprica	2	2	43.68263197	0
Anders	2	2	52.43511486	0
Leoben	2	2	28.92223597	77.64051199
Anders	2	4	101.5508718	55.41879892
Caprica	0	2	63.87834907	103.3684309

7. CONCLUSION

Multi-robot patrol is becoming increasingly applicable, and much work has been done in examining the efficiency and frequency with which the robots patrol. Patrolling along an open polyline creates a more challenging task in that the robots need to backtrack across the area to achieve even coverage. While high-frequency coverage is important, we chose to focus on maintaining

Table 2: Segment rotation using six robots to patrol four segments: 1, 2, 4, 5. Segment 3 is the base and is not patrolled.

Name	Segment	Passes	Total Passes
Leoben	4	4	4
Anders	1	10	10
Caprica	2	8	8
Hera	5	12	12
Sharon	4	11	11
Anders	4	0	10
Leoben	2	14	18
Anders	5	3	13
Tory	4	1	1
Caprica	1	20	28
Leoben	2	2	20
Tory	5	0	1
Sharon	5	4	15

the patrol long-term. The robots patrol their segments but remove themselves once their battery level gets below a certain threshold. It returns to base, and the remaining robots dynamically relocate to fill the empty space. Lastly, a freshly charged robot is released to reinforce the active robots.

Much can be done in future work. Currently, to cover all the segments at all times, there need to be enough working robots. An idea for future work would be to adapt the algorithm such that the robots could cover more than one segment if there are too few robots available. They could return to a single segment if a new robot were to arrive.

In addition, we are only patrolling equal length straight lines. Extending the algorithm to a more diverse setting would return more realistic results. Varying the shape of the line would require the segments to vary in length as the robots would need more time to patrol a more difficult region. It would also determine how adaptable the algorithm is to different environments.

8. ACKNOWLEDGEMENTS

This research was supported by the Distributed Research Experiences for Undergraduates (DREU) program through the Computer Research Association's Committee on the Status of Women in Computing Research (CRA-W). I would like to thank my mentor Dr. Maria Gini as well as the other students in the artificial intelligence and robotics labs for their support. In particular, I would like to thank Elizabeth Jensen and Mike Franklin for their part in this project.

9. REFERENCES

[1] N. Agmon, N. Hazon, and G. A. Kaminka The giving tree: constructing trees for efficient offline and online multi-robot coverage In *Annals of Mathematics and Artificial Intelligence,* vol. 52, pages 143-168, 2008

[2] N. Agmon, S. Kraus, and G. A. Kaminka Multi-Robot Perimeter Patrol in Adversarial Settings *Robotics and Automation, IEEE,* 2008

[3] H. Choset Coverage for Robotics–A Aurvey of Recent Results In *Annals of Mathematics and Artificial Intelligence,* vol. 31, pages 113-126, October 2001

[4] Y. Elmaliach, A. Shiloni, and G. A. Kaminka A Realistic Model of Frequency-Based Multi-Robot Polyline Patrolling In Padgham, Parkes, Müller, and Parsons, editors, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008),* Estoril, Portugal, 12-16 May 2008.

[5] Y. Elmaliach, N. Agmon, and G. A. Kaminka Multi-robot area patrol under frequency constraints. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-07),* 2007

[6] I. Rekleitis, A. P. New, E. S. Rankin, H. Choset Efficient Boustrophedon Multi-Robot Coverage: an algorithmic approach In *Annals of Mathematics and Artificial Intelligence,* vol. 52, pages 109-142, 2008