Comparison of the Effectiveness of Simple Agent Capabilities for an On-line Area Coverage Task

David Buckingham
Department of Computer Science
Tufts University
Medford, MA 02155, USA
dbucki01@tufts.edu

Giordano Ferreira
Department of Computer Science
Tufts University
Medford, MA 02155, USA
giordano.ferreira@tufts.edu

Alexander Bock
Department of Computer Science
Tufts University
Medford, MA 02155, USA
alexander.bock@tufts.edu

Matthias Scheutz
Department of Computer Science
Tufts University
Medford, MA 02155, USA
matthias.scheutz@tufts.edu

***Abstract*—We present the results of a suite of experiments conducted with a 2D simulation of a multi-agent area coverage problem. Agents perform a basic behavior of moving in a straight line and responding to collisions by setting a new random heading. Agents are optionally equipped with a subset of three additional capacities: a timer that causes the agent to alter its heading at regular intervals, a surface sensor that detects areas that have not yet been covered, and an agent sensor that detects the proximity of other agents. Our experimental conditions included all combinations of these features and also varied the task environment and the number of agents deployed. We found that the usefulness of feature configurations depends upon the task environment, number of agents deployed, and the metric used to determine performance. The surface sensor feature became more useful as the number of agents increased, while the agent proximity sensor became less useful. The surface sensor was generally useful early in the simulation, but became a hindrance once the task was nearly complete. With one performance metric, each of the three features alone was detrimental, while a configuration combining them together was beneficial.**

***Index Terms*—simple agents, area coverage, multi-agent system**

I. Introduction

The area coverage problem requires one or more agents to move around a task environment in order to pass an effector over all points within that environment. As an agent moves, its effector causes previously uncovered area to become covered area. Domains such as search and rescue tasks, autonomous painting, and military applications may benefit from using approaches to the area coverage problem that use groups of robots with minimal sensory and computational capabilities. Despite their limitations with respect to more powerful and complex robots, the potential advantages of such *simple* robots include small size, low cost, disposability, increased system robustness, and simplified logistics. According to the definition proposed by Wagner et al. [23], a simple robot 1) has memory capacity that is independent of the problem size or the number of robots; 2) can sense only a small part of the environment; 3) has limited computational resources; and 4) rarely uses explicit communication.

Area coverage problems are relevant for a large set of real world domains such as floor cleaning [16], [21], industrial panting [2], [6], agriculture [11], [15] and demining [1], [18]. Such tasks can be approached using diverse algorithms depending on the types of agents and the information available, including path-planning algorithms when *a priori* task information is complete. Randomized approaches have been successfully employed in several commercial floor cleaning robots, even though they do not guarantee completion [16]. Galceran and Carreras argue that randomized approaches may not effectively scale for vast areas because of redundant exploration [9]. Yet, randomized approaches have the advantage that they require few or no sensors for localization, and they can be implemented in robots with limited computational resources. Therefore, we argue that randomized control strategies may be appropriate for simple robots. Even though randomized approaches do not guarantee complete coverage, a group of simple agents may perform sufficiently well to satisfy some reasonable performance constraints.

Choset [7] proposes a taxonomy for area coverage algorithms. The first distinction is between heuristic and complete algorithms based on whether they guarantee completion given some constraints. A second distinction is between on-line and off-line methods. Off-line methods rely on complete information about the target area. While off-line approaches may be feasible in some circumstances, complete a priori knowledge may not always be available. In these cases, robots must use their sensors to acquire information about the environment. Although some on-line methods are complete, none are optimal for all environments.

Another important characteristic is the number of agents working on the task. While increasing the number of agents can reduce the total time to complete the task by reducing the amount of work for each agent, Rekletis et al. [19] emphasize that this performance gain occurs only if the workload can be

978-1-5386-9276-9/18/$31.00

effectively divided to minimize areas covered multiple times.

We present a novel systematic comparison of all combinations of several possible simple agent features. Our previous work with the area coverage problem [5] did not qualitatively vary agent capacities: agents were equipped with the equivalent of this work's *timer* feature, but lacked sensors. Our current simulation further improves upon our previous work by including inter-agent collision detection.

Our simulation is a spatially continuous 2D implementation of the area coverage problem. A group of 10, 20, or 40 agents deployed in the simulation share the same features and abilities. Agents have no prior knowledge of the task environment: the layout of the target area can affect their behavior only by means of sensors and on-line interactions. Thus, according to Choset's taxonomy, our method is a randomized, on-line, homogeneous multi-agent approach to the area coverage problem.

Some approaches to the area coverage task in multi-agent systems consist of agents working as a team, communicating often to decide the next action for each robot. Hence, direct communication has an important role in the task performance of the group [8]. In other approaches, each robot uses information left in the environment by other agents (indirect communication). In our simulation, each robot autonomously decides its next action, i.e. there is no group decision.

Many real-world robotics tasks must contend with uncertainty, e.g. caused by errors in path tracking, noisy sensor data, or motor imprecision. Paull et al. [17] offer an approach to area coverage that accounts for uncertainty by modeling coverage probabilistically. Their method guarantees complete coverage without assuming bounded state error. Because of our focus on agents with limited computational resources, we do not simulate uncertainty. However, future work could explore how uncertainty influences the area coverage performance of simple agents.

One advantage of using multiple robots is increased robustness of the system. This was the focus of the algorithm proposed by Hazon, Mieli and Kaminka [10]. The authors present a complete multi-agent algorithm for the on-line area coverage task. They prove that the algorithm is robust against the failure of individual robots, and guarantees completeness as long as one robot is still functioning. However, their approach relies upon agents using sensors to create an internal representation of the task environment, making it unsuitable for simple agents.

Our agents perform a basic move forward behavior: they move in a straight line and respond to collisions by randomly setting a new heading. Agents were also optionally equipped with a subset of three additional features. 1) A *timer* causes an agent to set its course at regular time intervals. 2) A *surface* sensor allows an agent to detect and move towards nearby areas that have not yet been covered. 3) A *proximity* sensor allows an agent to detect and move away from other nearby agents. We selected these agent features because they correspond to capacities available to many simple robots. The r-one robot [13], for example, has an onboard clock, an omnidirectional bump sensor to respond to collisions, and an infrared obstacle detector that might be used to detect other robots. A camera could detect surface area that has not yet been altered by an effector (e.g. unpainted surface). The e-puck robot [4] has similar capabilities.

We present our agent-based simulation of the area coverage task in Section II. Then, in Section III, we present the experiments we performed to compare the combinations of features for two distinct environments and various numbers of agents. In Section IV, we present the results of our experiments and examine how each feature combination affects performance. We discuss the significance of our results in Section V, and conclude the paper with Section VI by reviewing our findings and proposing directions for future work.

II. Area Coverage Simulation

Each circular agent in our simulation has three degrees of freedom: two translation components and one component of orientation. As an agent moves, its circular effector remains centered over the center of the agent, causing any uncovered area under the effector to become covered area. Although agent movement is computed in discrete timesteps, the Minkowski sum of the effector across sequential timesteps becomes covered area, simulating the continuous movement of the effector between its positions at sequential timesteps.

We use the radius of an agent's body, r, as the fundamental metric of distance for our simulation. This corresponds to 35mm for an e-puck robot. Tab. I specifies several measurements in terms of r.

TABLE I: Simulation parameters in terms of agent body radius

Agent radius	$1 \cdot r$
Effector radius	$1.414 \cdot r$
Surface sensor range	$4 \cdot r$
Proximity sensor range	$10 \cdot r$
Agent speed	$0.\overline{3} \cdot r$
Simple arena area	$100 \cdot r^2$

A. "basic" *configuration*

At each timestep t an agent has a position, $\langle x_t, y_t \rangle$, and a heading, which can be described as an angle θ_t or as a unit vector $(h_{xt} h_{yt})$, where $h_{xt} = \arccos(\theta_t)$ and $h_{yt} = \arcsin(\theta_t)$. At each timestep, an agent's position is updated according to its heading such that $x_{t+1} = x_t + (h_{xt} \cdot s)$ and $y_{t+1} = y_t + (h_{yt} \cdot s)$, where s is the agent speed.

When such movement results in a collision, causing the agent's body to overlap with a wall or another agent, the collision detection and resolution algorithms of the *Chipmunk2D* [22] physics library moves the agent away from the overlapping region until there is no overlap. At the start of the simulation and after each collision an agent's heading is set to a new direction drawn uniformly from $\{0, 2\pi\}$. Thus, in the *basic* configuration, agents move in straight lines in random directions and set a new random heading after collisions.

Agents are equipped with one of 8 feature sets, corresponding to the power set of three optional features: *timer*, *surface*,

and *proximity*. When one or more features is present they augment the basic behavior described above. The collision response behavior of the *basic* configuration is preserved in all configurations.

B. "timer" *configuration*

Every 50 timesteps the agent sets a new heading drawn uniformly from $[0, 2\pi)$. We chose a 50 timestep duration based on visual observation of the simulation to allow agents to move a reasonable distance across the environment (15%) before changing direction.

C. "surface" *configuration*

In the *surface* configuration, the agent uses a sensor that measures covered area to calculate a new heading. Each timestep (except for immediately after a collision), 16 equally spaced lines of length $4 \cdot r$ are drawn radiating from the center of the agent (like wheel spokes). For each line, the total length intersecting non-covered area is calculated. The agent sets a new heading in the direction of the line that intersects the most uncovered area. If there is a tie, one of the lines intersecting the most uncovered area is randomly selected. Thus, if no lines intersect any covered area, the agent uniformly draws a new heading from $0, \pi/8, \pi/4, \ldots, 2\pi$.

D. "proximity" *configuration*

In the *proximity* configuration, the agent uses a sensor that measures the direction of nearby agents to calculate a new heading. Each timestep (except for immediately after a collision), if there is at least one other agent within $10 \cdot r$, set a new heading directly away from the nearest agent. Otherwise, the heading is not changed.

E. "surfaceProximity" *configuration*

In the *surfaceProximity* configuration, each timestep, if there is at least one other agent within $10r$, the heading is updated as in the *proximity* configuration. Otherwise, the heading is updated as in the *surface* configuration.

F. "timerSurface" *configuration*

In the *timerSurface* configuration, the agent uses the covered area sensor to update its heading as in the *surface* configuration. However, it does so only every 50 timesteps.

G. "timerProximity" *configuration*

In the *timerProximity* configuration, the agent uses the agent sensor to update its heading as in the *proximity* configuration. However, it does so only every 50 timesteps.

H. "timerSurfaceProximity" *configuration*

In the *timerSurfaceProximity* configuration, the agent updates its heading as in the *surfaceProximity* configuration. However, it does so only every 50 timesteps.

III. Experiments

The simulation stores each agent's coordinates in the environment. The simulator also stores the ratio of the covered area to the total environment area. The simulation halts when either that ratio is 1 (the whole environment is covered) or 100,000 timesteps have elapsed.

We used two environments which have appeared in previous work using randomized approaches to the area coverage task [5]. The first environment, called *Simple*, is an empty square with edges of length $100r$. The second environment called *Walls*, was first proposed by Wagner et al. [24]. This more complex environment requires agents to navigate two distinct areas, linked only by a tight corridor, which must also be covered. The outer bounds have length of $100r$, the top area contains 25% of the environment area, the bottom area is 60% of the entire area, and the corridor plus inner walls contain the other 15%. Fig. 1 illustrates the two environments.

Note that the area to be covered in the walls environment is 9.5% smaller than the simple environment. Thus while agents must contend with the barriers in the *walls* environment, the total area to be covered is smaller than in the *simple* environment.

The first parameter we varied was the number of robots in the environment $n \in \{10, 20, 40\}$. Increasing the number of robots improves the overall performance of the group. However, the improvement is dependent on how well the robots can divide the task. Simple agents do not communicate or maintain an explicit representation of the environment, making it harder to cooperate. Thus, we expect the improvement to asymptote as the number of robots increase, and using a large number of robots may not be justifiable.

Fig. 1: The $Simple$ and $Walls$ environments.

With three optional features, there were 8 unique feature combinations. For each experimental condition consisting of a combination of features, a number of robots in $\{10, 20, 40\}$, and one of the two environments, we ran 50 simulations with randomized starting positions. Thus, there were 2400 experimental runs in total.

To evaluate the performance of the robots, we used two metrics. The *ratio completion* metric, R, expresses how many timesteps it took to cover some ratio of the environment. We recorded values for 5 ratios: 0.5, 0.9, 0.95, 0.99, and 1. Thus, $R_{0.5}$ is the number of timesteps elapsed when first half of the environment becomes covered, R_1 is the how many timesteps it takes to cover the entire environment, etc.

TABLE II: Number of simulation runs that achieved each ratio completion metric for experiment conditions that did not complete the area coverage task for all 50 runs.

Task	Configuration	#	50%	90%	95%	99%	100%
simple	surfaceProximity	10	50	50	50	50	**35**
simple	surfaceProximity	20	50	50	50	50	**48**
simple	surface	10	50	50	50	50	**36**
simple	surface	20	50	50	50	50	**43**
walls	surfaceProximity	10	50	50	**49**	**49**	**13**
walls	surfaceProximity	20	50	50	50	50	**34**
walls	surfaceProximity	40	50	50	50	50	**44**
walls	surface	10	50	50	50	**49**	**10**
walls	surface	20	50	50	50	**48**	**41**
walls	surface	40	50	50	50	50	**46**
walls	timerSurface	10	50	50	50	50	**49**
walls	timer	10	50	50	50	50	**49**

For most experimental conditions, the area coverage task was completed before the 100,000 timestep limit, and all 50 runs were used to calculate each ratio completion metric median and distribution. For runs that did not complete the task, R_1 is undefined, and some runs didn't even achieve some of the lesser ratio thresholds. Tab. II shows, for experimental conditions that did not completely cover the environment before reaching the timestep limit in all 50 runs, how many of the runs reached each ratio threshold, and therefore how many runs are represented in our result statistics.

The *integral* metric, I, is the sum of the covered ratio at each timestep over all 100,000 timesteps. Thus, if $r(t)$ is the ratio of covered area to total environment area at timestep t, then $I = \sum_{t=1}^{100000} r(t)$. If the simulation ends before the timestep limit because $r(u) = 1$ at some timestep $u < 100000$, then the remaining timesteps are still included in the metric; that is, $I = 100000 - u + \sum_{t=1}^{u} r(t)$.

Note that better performance is indicated by lower values of the ratio completion metric, but by higher values of the integral metric.

IV. Results

A. Comparing the number of agents

Fig. 2 shows performance using the integral metric for each agent configuration and environment, with separate plots for each number of agents. Dots show mean values across 50 experimental runs and vertical lines show 95% confidence intervals. We will consider how the relative performance of each feature combination varies with the number of agents.

With 10 agents, the performance of the *surface* configuration was similar to that of the *basic* configuration. With 40 agents, however, *surface* outperformed *basic*. With 20 agents, the effect of *surface* depended on the environment: it helped in *simple* and hurt in *walls*. For all numbers of agents and environments, *surface* resulted in broader performance distributions than *basic*.

This relationship between number of agents and effect on performance was roughly reversed for the *proximity* configuration. With 10 agents the performance of the *proximity* configuration was similar to *basic*, but with more agents the *proximity* configuration was worse than *basic*.

With 10 agents, *timerSurface* was better than just *surface*, but with 40 agents *surface* alone was better than *timerSurface*.

Overall, the performance distribution were tighter in *simple* than in *walls*: the randomized starting position of the agents had a greater bearing on performance in *walls*. If, for example, all agents start on one side of the barrier, it might take a long time before any area on the other side of the barrier becomes covered. For most conditions, this difference was more pronounced with fewer agents.

B. Comparing performance metrics

Fig. 3 shows performance distributions for each agent configuration and environment, aggregating the different numbers of agents, and using separate plots for each ratio covered metric. Thick horizontal lines show median values across multiple experimental runs (50 runs except for as reported in Tab. II), colored boxes show one interquartile range (IQR), whiskers show 1.5 IQR, and dots show results beyond 1.5 IQR.

The *surface* configuration helped performance slightly for the $R_{0.5}$, $R_{0.95}$, and $R_{0.99}$ metrics, but harmed performance for the R_1 metric. It increased the speed with which agents covered new area until after 99% of the arena had been covered. After that, however, performance was much worse than with the basic behavior.

For the R_1 metric, *timer*, *surface*, and *surfaceProximity* performed worse than the basic behavior. The *surface* feature hurt performance in all cases except where it was combined with both *proximity* and *timer*. While the *timer* feature decreased performance in isolation, it's addition to the *surface* feature, with or without the *proximity* feature, helped performance.

For the $R_{0.99}$ metric, *timerSurfaceProximity* had lower medians and tighter distributions than the basic behavior in both environments. In the *simple* environment, each feature in isolation hurt the median performance, increasing the number of timesteps to cover 99% of the environment, but improved performance in combination.

V. Discussion

The *surface* feature improved median performance over *basic* in the *simple* environment only when the number of agents was at least 20, and in *walls* only with 40 agents. This feature provides a degree of indirect coordination between agents: an agent can detect and avoid areas that have already been covered by other agents (or by the same agent). As previously discussed, increasing the number of agents can only improve performance to the extent that redundancy is avoided by a division of the task among agents. The improvements to performance caused by the task-partitioning quality of the *surface* feature may therefore increase as the number of agents (and the amount of redundancy in the *basic* configuration) increases.

The *proximity* feature helped performance with few agents but hurt performance with many agents. Visual observation of

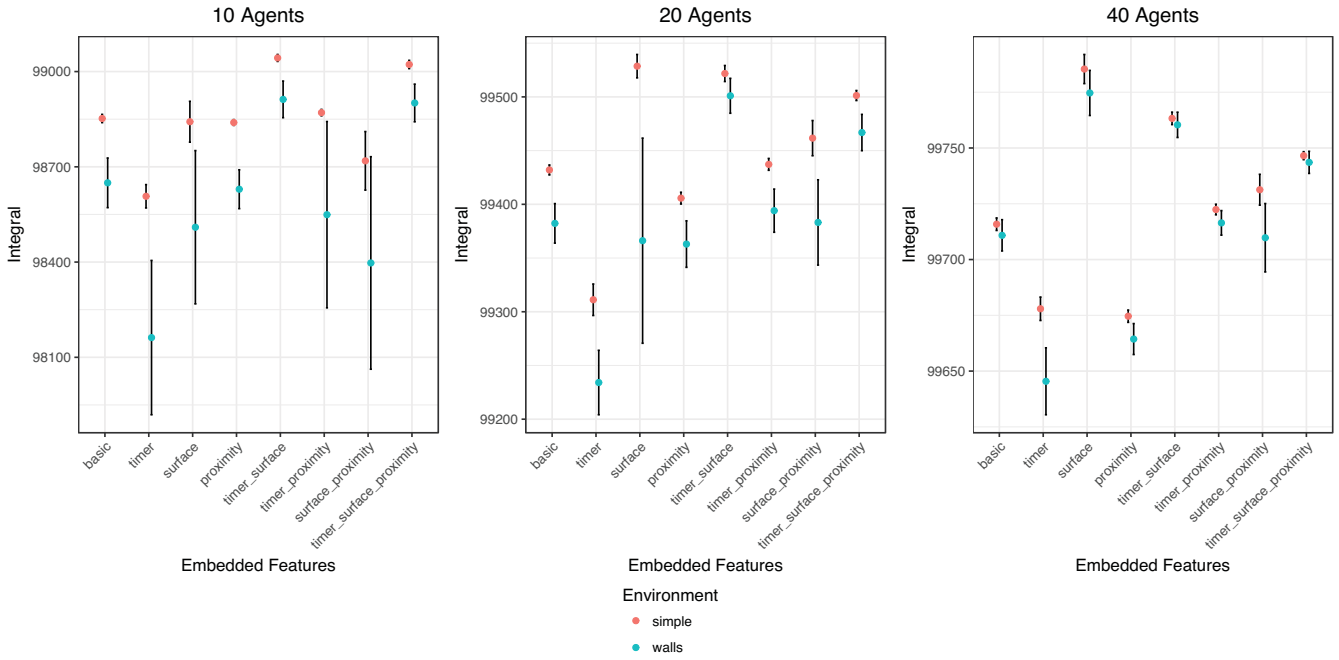


Fig. 2: Mean performance and 95% confidence intervals for all agent configurations with 10, 20, and 40 agents using the integral metric.

the simulation with the *proximity* configuration suggests that large populations of agents tend to stay around the outer edges of the task environment. Their mutually repulsive behavior prevents many agents from moving about the center, and the task of covering much of the environment falls to relatively few agents. Note that the *timer* feature negated this phenomenon: and *timerProximity* outperformed *basic* with 20 and 40 agents. With less frequent sensing and heading alterations, the agents do not avoid the center of the environment.

With 10 agents, *timerSurface* was better than *surface*, but with 40 agents *surface* alone was better than *timerSurface*. We hypothesize that with few agents, long periods of straight movement decreased each agent's redundant movement over areas that it had already covered, while with a larger population such straight movement caused agents to re-cover areas that had already been covered by others. In the 40 agent case, more frequent use of the sensor (every timestep in the *surface* configuration) might better help agents avoid areas already covered by others.

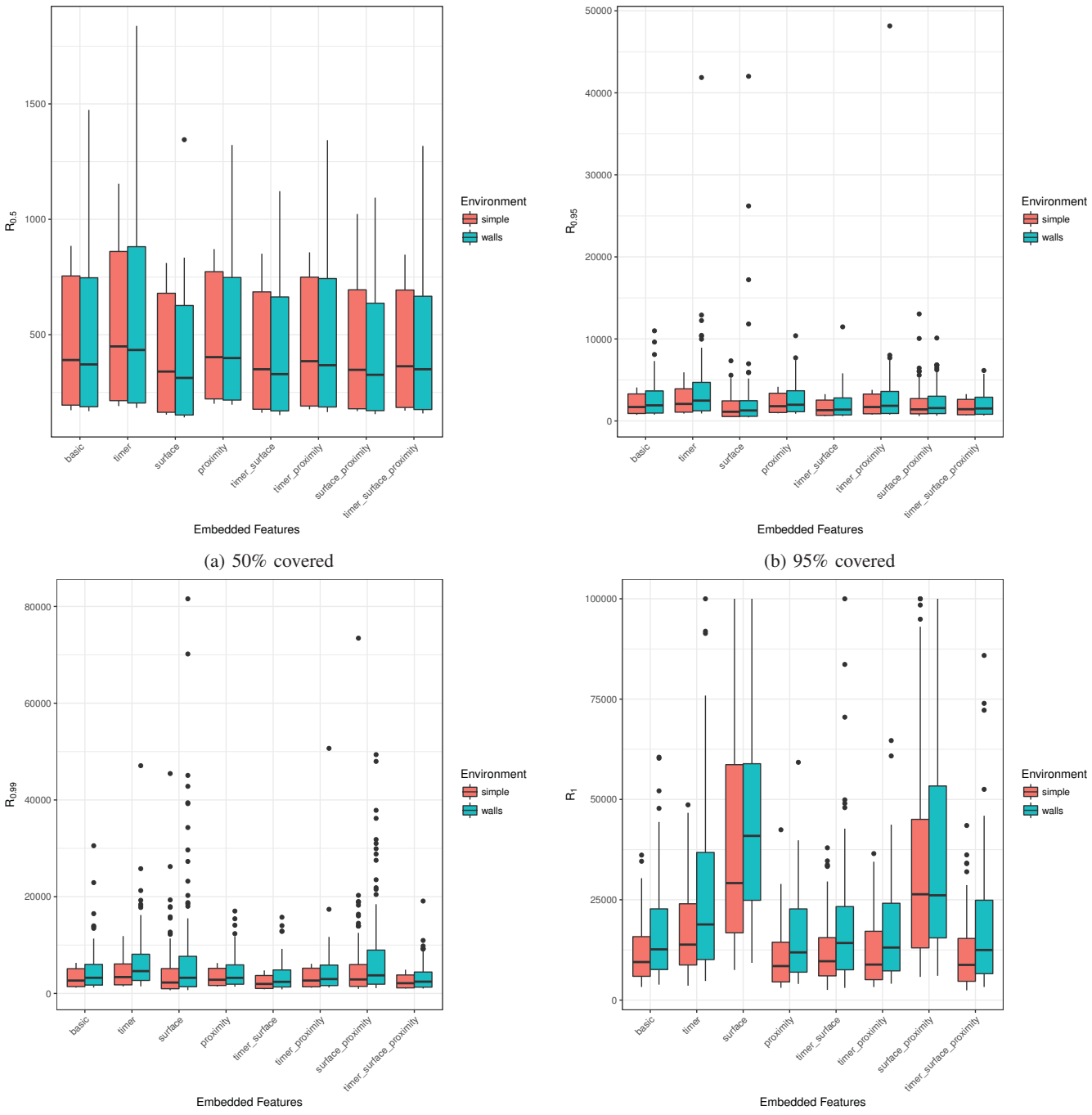
The *surface* configuration decreased performance according to the R_1 metric, as compared to *basic*, but not for any of the other ratio completion metrics. This suggests that use of the *surface* sensor increased the speed of area coverage until after 99% of the arena had been covered, but made it much harder for agents to find any small remaining uncovered areas. With the *basic* behavior, agents move in straight lines, only changing direction after collisions. With the arena nearly covered, all of the raycasts of the *surface* feature will usually intersect no uncovered area, and the agent will select a new direction randomly from the 16 regularly spaced ray directions at each timestep. It seems that this behavior is less effective at covering the last, small uncovered area than the basic behavior.

For the R_1 metric, i.e. considering how long it took to complete the area coverage task, the *timer* feature hurt performance in isolation, but it's addition to the *surface* feature, with or without the *proximity* feature, helped performance. Thus, randomly altering direction at set timesteps was worse than just going straight, but applying the surface sensors to determine a new direction at set timesteps was better than doing so every timestep.

Perhaps our most interesting result is seen in the *simple* environment with the $R_{0.99}$ metric. The *timerSurfaceProximity* configuration had better median performance and a tighter distribution than the basic behavior, even though each feature in isolation (the *timer*, *surface*, and *proximity* configurations) had worse median performance than *basic*. Thus, the features were detrimental alone, but beneficial together.

Most configurations performed slightly better in *walls* than in *simple* for the $R_{0.5}$ metric, but the reverse is generally true for the other metrics. The *walls* environment is slightly smaller than *simple*, even though the barrier in *walls* makes it, in some sense, more difficult. Our agents tend to cover half of the *walls* environment faster than they cover half of the *simple* environment, but after that, the barrier in *walls* slows down progress as compared to *simple*.

Overall, our results suggest a trajectory of improved performance with increasing numbers of agents. We would expect this tendency to continue with even greater numbers of agents, but with diminishing returns as redundant area coverage also increases. We also expect the usefulness of the *proximity*



(a) 50% covered

(b) 95% covered

(c) 99% covered

(d) 100% covered

Fig. 3: Tukey boxplot showing median performance and distributions for each agent configuration using each ratio completion metric.

feature to continue to diminish as the number of agents increases.

In previous work [5] we have emphasized the importance of analyzing results distributions instead of just average performance in randomized approaches to the area coverage task. We suggested that considering result distributions can help robot designers to specify constraints to the performance distribution and then rule out configurations with poor performance. As in those findings, our present results illustrate that the importance of starting position depends greatly upon the environment: our performance distributions across multiple starting placements are much wider in *walls* than in *simple*. It is also noteworthy that the relative performance of agent configurations differs for the two task environments. Future work should establish a tunable metric of environment complexity to determine how such complexity affects performance.

Batalin and Sukhatme [3] proposed two behavior-based approaches based on the premise that robots must spread out over the environment to achieve proper coverage. One of their approaches (called "Molecular") is similar to our *timerProximity* configuration. They used visual communication to avoid crowded areas and a simple memory mechanism, called "timeout". Our results confirmed their findings about the usefulness of avoiding crowded areas as *timerProximity* outperformed *basic* for 20 and 40 agents in the *walls* environment and for all numbers of agents in *simple*.

Given the importance of considering the cost of equipping agents with different capacities in real-world applications [20], suboptimal configurations might be preferable to the best-performing configurations for some tasks. With 10 agents, for example, *timerSurface* outperformed *surface* in both environments, but the difference was greater in *walls*. The added benefit of the *timer* feature might outweigh its cost in the *walls* environment but not in *simple*. As another example, for all numbers of agents, *timerSurfaceProximity* outperformed *timerProximity*. Depending on the cost of *surface*, the inclusion of that feature might or might not be worthwhile.

We propose that future work employ some simple heuristics that might help performance in the area coverage task without greatly increasing agent complexity. For example, a "spiral" movement pattern could be implemented with a timer mechanism with a delay that increases over time. A covered area sensor could be used to perform an edge-following behavior. Simple inter-agent communication capacities could be used to generate formations that coordinate movement and avoid redundant covering.

Future work should also compare the performance of simple randomized methods with more complex approaches to the area coverage problem. For example, Wagner et al. [23] proposed a bio-inspired algorithm for area cleaning based on ant behavior. In that work, robots treat the dirt on the floor as an indirect form of robot-robot communication. Rañó et al. [14] used the chaotic behavior of a Braitenberg vehicle to solve the area coverage problem, showing that such behavior is equivalent to the movement of charge in an electromagnetic field. Luo et al. [12] proposed a multi-agent on-line approach where each neural network-controlled robot treats other robots as moving obstacles. These approaches require more complex agents than those presented in this work. Therefore, it may be necessary to define a standard metric of cost and calculate a performance measure for each method that takes cost-efficiency into consideration. For example, [20] found that inter-agent communication was beneficial in absolute terms under specific conditions in a multi-agent territory exploration task, but that the benefit was only worthwhile if the cost of communication was low. A simple *wait timer* mechanism, comparable in complexity to our *timer* feature, provided performance benefits equivalent to communication and presumably could be implemented at much lower cost.

We chose our set of agent features because they correspond to the real capacities of some simple robots, are easily implemented in simulation, and are few enough to allow consideration of all feature combinations. Future work can expand the set of sensors (e.g. a wall/obstacle sensor) and capacities (e.g. remembering covered areas) available to agents.

Our long term goal is to test different approaches to the area coverage task in the real world. However, the results of real world behavior are less predictable than in our simulation. For example, the effector might leave small areas uncovered, requiring a second covering. In addition, covered areas can become uncovered after some time. For example, a floor can become dirty again after a robot cleans it. Our future work will pursue tasks that are more realistic. Using more complex environments, new heuristic approaches, and more robot configurations, we can consider questions such as "Can a group of robots maintain a necessary coverage ratio indefinitely even as previously explored areas revert to unexplored status?" and "Do randomized methods sufficiently scale for vast areas compared to more complex heuristic approaches?"

VI. Conclusion

We have systematically compared the performance of simple agent configurations in a simulated multi-agent on-line area coverage task. We tested all combinations of 3 agent features in two distinct environments and for different numbers of robots. Our features consisted of a simple timer (*timer*), a covered area sensor (*surface*), and an agent proximity sensor (*proximity*).

We found that the usefulness of feature configurations depends upon the task environment, number of agents deployed, and the metric used to quantify performance. The main conclusions we have drawn from our results are:

- Agents that avoided each other tended to stay around the outer edges of the task environment, decreasing performance.
- Long periods of straight movement decreased redundant coverage with few agents, but increased it with many agents.
- The *surface* feature becomes more useful as the number of agents increases.

- The *proximity* feature became less useful as the number of agents increased.
- The *surface* feature improved area coverage only until the environment was almost completely covered.
- Randomly altering direction at set timesteps was worse than just going straight.
- Applying the surface sensors to determine a new direction at set timesteps was better than doing so every timestep.
- In the *simple* environment with the $R_{0.99}$ metric, the agent features were each detrimental independently, but beneficial in combination.

Using our integral metric, which incorporates performance over the entire simulation run, we found that the best configurations for 10 agents were *timerSurface* and *timerSurface-Proximity*. With 20 or 40 agents, the best configuration was *timerSurface*.

REFERENCES

[1] Ercan U. Acar, Howie Choset, Yangang Zhang, and Mark Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International Journal of Robotics Research*, 22(7-8):441–466, 2003.

[2] Prasad N. Atkar, Aaron Greenfield, David C. Conner, Howie Choset, and Alfred A. Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.

[3] Maxim A. Batalin and Gaurav S. Sukhatme. *Spreading Out: A Local Approach to Multi-robot Coverage*, pages 373–382. Springer Japan, Tokyo, 2002.

[4] M Bonani. Raemy x. pugh j. cianci c. klaptocz a. magnenat s. zufferey j.-c. floreano d. mondada, f. and a. martinoli. the e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 59–65, 2009.

[5] David Buckingham, Giordano B. S. Ferreira, and Matthias Scheutz. Better than average: analyzing distributions to understand robot behavior in a multi-agent area coverage scenario. In *Proceedings of 14th European Conference on Artificial Life*, 2017.

[6] Heping Chen, Ning Xi, Zhouhua Wei, Yifan Chen, and Jeffrey Dahl. Robot trajectory integration for painting automotive parts with multiple patches. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3984–3989. IEEE, 2003.

[7] Howie Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, Oct 2001.

[8] Deepanwita Das, Srabani Mukhopadhyaya, and Debashis Nandi. Techniques in multi-robot area coverage: A comparative survey. In *Handbook of Research on Design, Control, and Modeling of Swarm Robotics*, pages 741–765. IGI Global, 2016.

[9] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258 – 1276, 2013.

[10] N. Hazon, F. Mieli, and G. A. Kaminka. Towards robust on-line multi-robot coverage. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1710–1715, May 2006.

[11] Alireza Janani, Lyuba Alboul, and Jacques Penders. Multi-agent cooperative area coverage: Case study ploughing (extended abstract). In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1397–1398, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems.

[12] Chaomin Luo, Simon X Yang, and Deborah A Stacey. Real-time path planning with deadlock avoidance of multiple cleaning robots. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4080–4085. IEEE, 2003.

[13] James McLurkin, Andrew J. Lynch, Scott Rixner, Thomas W. Barr, Alvin Chou, Kathleen Foster, and Siegfried Bilstein. *A Low-Cost Multi-robot System for Research, Teaching, and Outreach*, pages 597–609. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[14] Iñaki Rañó and José A Santos. A biologically inspired controller to solve the coverage problem in robotics. *Bioinspiration and Biomimetics*, 12(3):035002, 2017.

[15] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[16] J. Palacin, T. Palleja, I. Valganon, R. Pernia, and J. Roca. Measuring coverage performances of a floor cleaning mobile robot using a vision system. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4236–4241, April 2005.

[17] Liam Paull, Mae Seto, John J. Leonard, and Howard Li. Probabilistic cooperative mobile robot area coverage and its application to autonomous seabed mapping. *The International Journal of Robotics Research*, 37(1):21–45, 2018.

[18] José Prado and Lino Marques. *Energy Efficient Area Coverage for an Autonomous Demining Robot*, pages 459–471. Springer International Publishing, Cham, 2014.

[19] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2):109–142, 2009.

[20] Paul Schermerhorn and Matthias Scheutz. Investigating the adaptiveness of communication in multi-agent behavior coordination. *Adaptive Behavior*, 15(4):423–445, 2007.

[21] G. Schmidt and C. Hofner. An advanced planning and navigation approach for autonomous cleaning robot operations. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 2, pages 1230–1235 vol.2, Oct 1998.

[22] Howling Moon Software. Chipmunk2d physics. https://www.chipmunk-physics.net, 2013 (accessed September 9, 2018).

[23] Israel A. Wagner, Yaniv Altshuler, Vladimir Yanovski, and Alfred M. Bruckstein. Cooperative cleaners: A study in ant robotics. *The International Journal of Robotics Research*, 27(1):127–151, 2008.

[24] Israel A. Wagner, Michael Lindenbaum, and Alfred M. Bruckstein. Mac versus pc: Determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *The International Journal of Robotics Research*, 19(1):12–31, 2000.