Robust Distributed Coverage using a Swarm of Miniature Robots

Nikolaus Correll and Alcherio Martinoli

***Abstract*— For the multi-robot coverage problem deterministic deliberative as well as probabilistic approaches have been proposed. Whereas deterministic approaches usually provide provable completeness and promise good performance under perfect conditions, probabilistic approaches are more robust to sensor and actuator noise, but completion cannot be guaranteed and performance is sub-optimal in terms of time to completion. In reality, however, almost all deterministic algorithms for robot coordination can be considered probabilistic when considering the unpredictability of real world factors.**

This paper investigates experimentally and analytically how probabilistic and deterministic algorithms can be combined for maintaining the robustness of probabilistic approaches, and explicitly model the reliability of a robotic platform. Using realistic simulation and data from real robot experiments, we study system performance of a swarm-robotic inspection system at different levels of noise (wheel-slip). The prediction error of a purely deterministic model increases when the assumption of perfect sensors and actuators is violated, whereas a combination of probabilistic and deterministic models provides a better match with experimental data.

I. Introduction

Multi-robot coverage [1]–[3] requires the coordination of a team of robots so that the environment is covered in the shortest possible time. Ideally, coverage is provably complete. Algorithmic performance and controller design is a function of the capabilities of the individual platform (sensors, communication devices) and the system as a whole (amount of a priori knowledge, centralized vs. decentralized approaches).

Regardless of the complexity of the chosen approach for coordination, the ability to perform a more or less crude cellular decomposition of the environment is the basis for most coverage algorithms [4] except those that are fully probabilistic [5]. For instance, there exist coverage algorithms that only require bumper sensors [6], and others that use long range sensors for decomposing the environment [3], [7]. Also, literature distinguishes between approaches that plan the robots' trajectories off-line [2], [8], and those that perform coverage on-line, in which case the environment needs not to be known in advance [3], [6], [7].

We study the multi-robot coverage problem using a case study concerned with covering the boundaries of elements aligned in a regular structure. Here, a team of robots is required to cover every point on the boundaries of all objects in an environment by its sensors. We initially tackled this problem by using a simple, randomized algorithm that used the structure of the environment as a template for coordinating the robot swarm [5], and analyzed it with probabilistic, macroscopic models. Moreover, a provably complete, near-optimal algorithm that assures coverage of all *edges* in a graph, where edges represent the boundary of objects in the environment and navigable routes between them, was proposed for the same case study in [8]. Whereas the randomized, template-driven approach *probabilistically* covers the environment without relying on global localization and centralized control, the deterministic approach requires perfect localization and navigation abilities, as well as off-line computation of the robots' trajectories for providing a near optimal and provably complete solution. Similarly, [1] uses a variant of the *bin packing* problem for covering all vertices of a graph by a robot team, and [2] provides a heuristic that leads to complete coverage if at least one robot does not fail. Little research, however, considers potential failures of the individual robots or their subsystems to execute parts of a deliberative control scheme in a probabilistic sense (see for instance [7]), and investigates algorithms with respect to the limitations of robotic systems operating under real-world conditions.

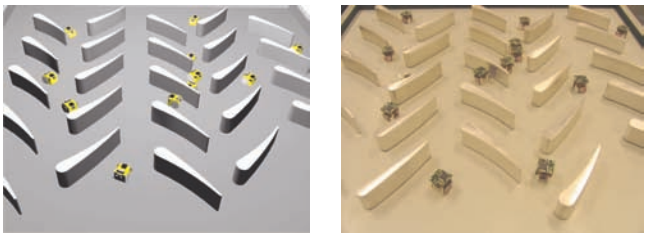

Fig. 1. *Left*: Overview of the turbine set-up in the realistic simulator. *Right*: Overview of the real-robot set-up.

In [9] we showed that the boundary coverage problem for structures with identical elements (Fig. 1) is equivalent to cover all *vertices* of a graph by a team of robots, which corresponds to coverage of all cells of a grid when the elements are aligned in a regular pattern (as it is the case in [1], [2], [10]). We used an algorithm that leads to complete coverage of an unknown grid by incrementally constructing a minimal spanning-tree, which is traversed by combining low-level reactive control with deliberative planning. Robots executed this algorithm in parallel without explicit collaboration. Although provably complete by design, the reactive behaviors that move the robots from element to element were prone to mistakes due to sensor and actuator noise and reduce the approach to probabilistic completeness.

Both authors are sponsored by a Swiss NSF grant (contract Nr. PP002-68647).

N. Correll and A. Martinoli are with the Swarm-Intelligent Systems Group (SWIS), École Polytechnique Fédérale Lausanne, Switzerland. nikolaus.correll@epfl.ch, alcherio.martinoli@epfl.ch

In this paper, we study this phenomenon systematically by first developing a microscopic deterministic model for distributed coverage, which assumes the robots behave perfectly (Section III-A). Motivated by real-robot experiments that do not fulfill this assumption [9], we consider a certain probability for the robots to reliably navigate from cell to cell (Section III-B), leading to a probabilistic model. We then use a realistic simulator *Webots* [11], to study the system's performance at different level of wheel-slip to validate our modeling approach.

II. Motivating Case study: Distributed Boundary Coverage of Regular Structures

The distributed boundary coverage problem has applications in various potential robotic tasks, such as inspection or maintenance of structures [8]. Our case study in particular is motivated by the inspection of the blades in the compressor section of a jet turbine, a process currently performed using borescopes, which is time consuming and costly [12]. The narrow structure of the turbine motivates the use of extremely miniaturized robots with limited capabilities.

We consider the environment as completely inspected when every single blade has been circumnavigated at least once. Sensor information collected during circumnavigation can then be stored and eventually broadcasted to a supervisor for analysis, which is beyond the scope of this paper.

A. Experimental Setup

A 60cm×65cm arena is populated with 25 blades in a regular pattern (Fig. 1), mimicking the rotor and stator blades of a jet turbine. Task performance is assessed using an overhead vision system [13], which monitors the boundary region of each blade rather than analyzing an individual robot's trajectory. The metric thus does not differentiate between coverage progress due to a deliberative or random policy.

The *Alice II* robot [14] has a size of $2 \times 2 \times 2$cm, a differential wheel drive that reaches speed of up to $4\frac{cm}{s}$, and four infrared proximity sensors for obstacle detection (up to 3cm). Because the 4MHz micro-controller provides only 368 bytes of RAM, we use a custom extension module [9], which extends the computational power of the *Alice* robot by an order of magnitude, and allows us to implement more sophisticated collective navigation algorithms. Because of its light weight (4g) and insufficient wheel-adhesion as well as the crude spatial resolution provided by only four sensors, accurate navigation involving turns is almost impossible even over short distances (a few centimeters).

In order to systematically study sensor and actuator noise, the experimental setup was reproduced in *Webots* [11], a realistic simulator that is able to accurately model the non-linear sensor characteristics of the *Alice* robot, including Gaussian noise on the sensors as well as wheel-slip. For this case study, *Webots* simulations allow us to collect results about 3 to 4 times faster than in real robot experiments.

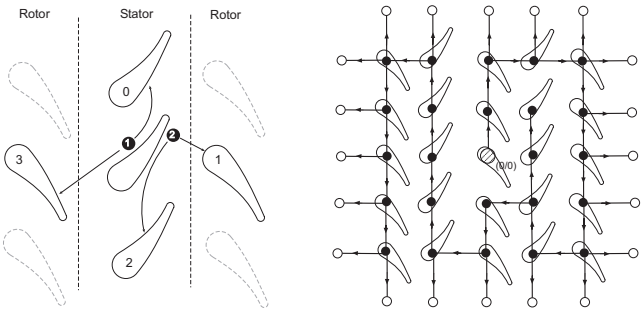

Fig. 2. *Left:* Way-points on a blade's boundary that can be navigated to using on-board sensors. *Right:* Possible trajectory for a single robot along a spanning-tree in a 5x5 blade environment (bold line). Backtracking paths are not shown.

B. Spanning-Tree Coverage

Exploiting the regularity of the environment for navigation and localization (by counting inter-blade transitions), the *Alice* can construct a spanning tree with the blades as vertices, and possible routes between a blade and its 4-neighborhood as edges (Fig. 2 for an example spanning tree). Edge traversal is achieved by a combination of dead-reckoning and navigation along way-points on a blade's boundary, (Fig. 2, *left*). Way-points can be distinguished by the robots on-board sensors that can detect a blade's tips as well as measure the curvature of the blade using odometry. Notice that a blade-to-blade navigation strategy increases the robustness of navigation on the spanning tree as the exact location at which a blade is hit does not matter.

The spanning tree is constructed on-line and systematically explored by a Depth-First-Search (DFS) algorithm. Edges are selected randomly from the set of unexplored edges — a policy that prevents robots from following the same trajectories once they met on a blade as well as promotes uniform distribution of the robots on the blade grid, and hence minimizes redundant coverage.

An edge of the spanning tree is considered fully explored when all nodes connected by it have been visited. Once all edges of a node are explored, the DFS algorithm makes the robot physically return to its parent node (known as *backtracking*)[1], and explores remaining unexplored edges of this node. The algorithm goes on until it reaches the spanning tree's root, a policy leading to provably complete coverage (see Theorem 1 below). Notice that the algorithm explores *all* possible edges, including those ending at a wall, leading to the creation of vertices that are actually not navigable. Fig. 2, *right*, shows a possible spanning tree constructed by the DFS algorithm; every edge will be visited twice (once on the way forth, and once on the way back to the root). Non-navigable vertices are indicated by circles.

C. Low-Level Reactive Robot Control

The spanning tree coverage algorithm requires the following low-level behaviors: obstacle avoidance, wall following,

[1]There exist spanning tree coverage algorithms that require no backtracking [10], but have higher demand on the robots' sensors, making them infeasible for the *Alice* robot

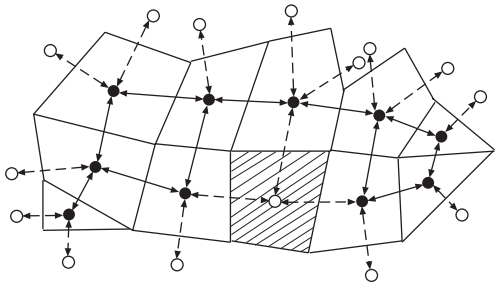

Fig. 3. An arbitrary environment with a cellular decomposition $\mathcal{V} \times \mathcal{E}$ consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. Dashed edges can only be partially navigated and dashed cells are obstacles.

assessing an objects type (blade, arena boundary, or another robot), determining the blade's type (rotor or stator) at the spanning tree's root, navigating to one of two distinct waypoints on a blade's boundary, traversing 8 possible edges (4 for rotor and 4 for stator blades), and finally backing up non-navigable edges (i.e., those ending in a wall). The DFS algorithm sequentially activates the appropriate behavior for physically guiding the robot along the spanning tree.

Robots that eventually stop following a policy leading to complete coverage due to sensor and actuator noise might still contribute to task progress and cover unexplored blades. For instance, a robot might miss a blade (wheel-slip, sensor noise), and continue exploration for a long time before it encounters a situation where its sensor readings do not match what it expects, e.g. when it encounters a wall instead a blade. Then a robot resets and starts over from scratch at its current location. Other reasons for failing are expiration of time-outs when trying to attaching to a blade, determining an objects type and so on, see [9] for more details.

III. A Microscopic Model for Distributed Graph Coverage

We will first ignore navigation errors due to sensor and actuator noise and assume the robots behave fully deterministically. We then model the algorithm described in Section II-B and prove its completeness. For a constant probability of failing at navigation leads, completeness becomes probabilistic, and we propose expressions for predicting the task progress as a function of the reliability of the individual robot.

A. Distributed Graph Coverage With Ideal Robots

We describe the cellular decomposition of the environment (in our case study the environment is discretized by the blades) as a directed graph $G = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$. Edges represent navigable routes between vertices and are bi-directional, that is they can be traversed by a robot in either direction. We refer to *neighbors* $\Gamma(v)$ of a vertex v as those vertices that are connected by an edge.

We assume that a robot is able to create such a representation of its environment online, thus the robot is able to perform a cellular decomposition of the environment as well as determine navigable paths between them solely using its on-board sensors while moving along the graph. For our miniature robot with minimal sensing capabilities, this is facilitated by the a priori known regular structure of the environment, whereas more powerful robots might be able perform cellular decompositions even of rough terrain. A sample environment with an arbitrary cellular decomposition is depicted in Fig. 3. We refer to the graph that is constructed by the robot when moving through the environment as *Spanning Tree* $\mathcal{C}(t)$. Upon complete exploration of $\mathcal{V}$, the Spanning Tree contains all vertices of $\mathcal{V}$; however only a subset of edges $\mathcal{E}$ have been physically traversed.

We denote $v \subseteq \mathcal{V}$ an individual vertex, and $\mathcal{E}^v = \{e_1^v, \ldots, e_{n_v}^v\} \subseteq \mathcal{E}$ the set of edges incident to v, where n_v are the number of edges connected to vertex v.

With every vertex of a robot's spanning tree $\mathcal{C}$, we associate a state space $\mathcal{X}^v = \mathcal{X}_c^v \times \mathcal{X}_p^v$, with $\mathcal{X}_c^v \in \{0, 1\}$ and $\mathcal{X}_p^v \in \{\emptyset, e_1^v, \ldots, e_{n_v}^v\}$. $\mathcal{X}$ is a 2-tuple that allows to keep track of whether a cell has been visited ($\mathcal{X}_c^v = 1$) or not ($\mathcal{X}_c^v = 0$), and which edge e_j leads to the vertex that has been visited prior to vertex v ($\mathcal{X}_p^v = e_j^v$). Initially, $\mathcal{X}_c^v = 0$ and $\mathcal{X}_p^v = \emptyset$ for all v. We will denote the vertex visited before vertex v as the *parent*, and the edge pointing to this vertex as the *parent edge of vertex* v.

We will now define a series of sets that allow us to formulate an expression for a robot's motion along the spanning tree. First, we define a set $\boldsymbol{N}$ that contains exactly one of the unexplored edges of vertex v

$$\boldsymbol{N}_v = f(\{e = (v, v') | \mathcal{X}_c^{v'} = 0\}) \tag{1}$$

$f(x) : \mathcal{E} \mapsto \mathcal{E}$, $x \subseteq \mathcal{E}$ is a condition that select one edge out of a set, e.g. $f(x) = min(x)$ might yield the edge with the lowest index, or $f(x) = rand(x)$ returns a random edge. $f(x)$ can be arbitrarily chosen, as long as it yields exactly one edge out of a set, and yields the empty set, if the set of edges is empty, that is $P(\emptyset) = \emptyset$. The notation $e = (v, v')$ denotes the edge e pointing from vertex v to vertex v'. By definition, $\boldsymbol{N}_v$ contains exactly one edge pointing from v to an unexplored neighbor of v, or is the empty set when all neighbors of v are explored.

Second, we define a set $\boldsymbol{P}$ that contains the edge pointing to the parent of a vertex v, and is the empty set, if v has any unexplored edges

$$\boldsymbol{P}_v = \{e \in \mathcal{E}^v | \mathcal{X}_p^v = e \wedge \forall v' \in \Gamma(v), \mathcal{X}_c^{v'} = 1\} \tag{2}$$

Finally, we define a set $\boldsymbol{V}$

$$\boldsymbol{V}_{v,e} = \{v' \in \Gamma(v) | e = (v, v')\} \tag{3}$$

that contains exactly one vertex that the edge e of vertex v points to.

Using the sets defined in equations (1)-(3), we can now define a recurrence equation that yields the next edge of a minimal spanning tree for a robot. Let be $p(t)$ the discrete position of a robot at time t, such that $\forall t : p(t) \in \mathcal{V}$. Using (1) and (2) we define the next edge the robot will move on as

$$e_{next} = \boldsymbol{N}_{p(t)} \cup \boldsymbol{P}_{p(t)} \tag{4}$$

Then, the vertex the robot will move on next is given by

$$p(t+\tau^{e_{next}}) = \boldsymbol{V}_{p(t),e_{next}} \tag{5}$$

$\tau^{e_{next}}$ is the time needed to to cross the edge e_{next} and cover the following vertex.

We can now define the state transitions for the vertex as

$$\begin{aligned} \mathcal{X}_c^{p(t+\tau^{e_{next}})} &= 1 \\ \mathcal{X}_p^{p(t+\tau^{e_{next}})} &= (p(t+\tau^{e_{next}}), p(t)),\ if\ \mathcal{X}_p^{p(t+\tau^{e_{next}})} = \emptyset \end{aligned} \tag{6}$$

and for the spanning tree as

$$\mathcal{C}\left(t+\tau^{e_{next}}\right) = \mathcal{C}\left(t\right) \cup p(t+\tau^{e_{next}}) \tag{7}$$

Theorem 1: For $p_i(t+\tau^{e_{next}}) = \emptyset$ all vertices in $\mathcal{V}$ have been explored, and robot i has returned to its initial position.

Proof: By definition of (2), a robot will never move to the parent of the current vertex it is on, if this vertex has still unexplored neighbors. Thus, (4) will always point towards unexplored vertices that are adjacent to its current position. Because this is true for all vertices, (4) yields the edge to the parent vertex only if the whole branch of the spanning tree connected by this edge has been fully explored. Finally, the spanning tree's root is the only vertex that does not have a parent, and thus (5) will yield the empty set. ■

We can now calculate minimal spanning trees by numerically solving (5)–(7) for an arbitrary set of vertices, edges, and initial placement of the robot on the graph. We extend this approach to multi-robot coverage by maintaining a spanning tree $\mathcal{C}_i$ as well as a state space $\mathcal{X}_i$ for every robot $0 \leq i < N_0$, with N_0 the total number of robots. For calculating the time to complete coverage, we iterate (5)–(7) until all vertices are covered.

Although we could achieve similar results with a point simulator, we notice that the proposed formalism allows for treating any cellular decomposition with arbitrary vertex shape and number of edges. Also, the formalism allows for describing possible explicit collaboration schemes among agents by operations on their spanning tree (not exploited in this paper). A MATLAB™ implementation and a series of sample environments is available on [15].

B. Modeling Sensor and Actuator Noise

We now associate a probability with every edge traversal. With the probability π^e to successfully traverse an edge e, we can calculate the probability to reach vertex v by the following recurrence equation

$$\pi_{i,v}(t+\tau^e) = \pi_{i,v}(t)\pi^e \tag{8}$$

where e is given by (4).

If π^e is constant for all edges, and Markovian, that is only dependent on the robot's current state, the probability to fail after traversing x edges follows a geometric distribution

$$P_{geo}(x) = (1-\pi^e)(\pi^e)^{x-1} \tag{9}$$

and the average number of vertices before failure μ calculates[2] to $\mu \approx \frac{1}{1-\pi^e}$.

[2]This is an approximation as the graphs usually have finite size.

When all robots have the same probability to fail, we can write the following recurrence equation for the average number of uncovered vertices after k trials, where one trial is considered to be the coverage of μ distinct vertices that are part of a minimal spanning tree constructed by a robot before it does a mistake that violates the completeness properties of its algorithm.

$$M_v(k+1) = M_v(k)\left(1-\frac{\mu}{\|\mathcal{V}\|}\right)^{N_0} \tag{10}$$

The length of one trial is given by $\mu\tau^e$, the average time needed for covering μ vertices. Equation (10) has a similar form as the model in [5], where a probabilistic model for random coverage of the environment is proposed: the likelihood of covering a virgin vertex decreases exponentially with the number of already covered vertices (compare also [7] for similar dynamics, although for a search task concerned with finding a fixed number of mines in a random fashion).

When $\mu = \|\mathcal{V}\|$, that is the whole graph is covered in one trial, (10) predicts exactly the same time to completion as the model for ideal robots ($\|\mathcal{V}\|\tau^e$). On the other hand, for $\mu = 1$, that is robots are unable to enforce the deliberative control policy, the system is equivalent to the implementation of [5].

IV. Results

Exemplar spanning trees that have been constructed by numerically solving (5)–(7) for random environments with obstacles are shown in the video accompanying this paper. All results are available on [15].

For validating our assumption that sensor and actuator noise leads to a time-independent (Markovian) probability to violate the completeness properties of a deliberative coverage algorithm when moving from blade to blade, we measure the number of blades a single robot can traverse without mismatch between its actual location on the spanning tree and its belief occuring in *Webots* for random drop-off locations and wheel-slip of 0.1% and 0.5% (6000 experiments each), as well as for real robots [9]. From this results (Fig. 4), we calculate the probability π^e of successfully traversing an edge using (9), the average time τ^e an edge traversal takes (including coverage of a vertex), and the average number μ of covered vertices before the completeness properties of the algorithm are violated (Table I). The first row of Table I represents values for ideal robots with best and worst division of labor among the robots (no redundancy and fully redundant).

We then measure the time it takes to completely circumnavigate every one of 25 blades in the arena of Fig. 1 for 1 to 10 robots and wheel-slip of 0.1% and 0.5% (100 experiments per team size and wheel-slip) using the controller described in Sections II-B and II-C. Notice that this policy requires the robots to start over once they are lost, as complete coverage would have been infeasible given the probability of successfully traversing an edge from Fig. 4. Results are compared with prediction of the model for ideal robots (5–7) for 1000 random drop-off locations of the team in Fig. 6,

Wheel-slip	π^e	μ^e	$\tau^e[s]$	Avg. nb. of vertices visited to completion	Time to completion for 10 robots [s]
0%	1	25	12	25–250	36–300
0.1%	0.79	4.77	38.5±10.3	63.21	541±252
0.5%	0.67	3.03	44±20.5	78.58	701±342
real	0.64	2.79	52±19.7	n.a.	788±375

TABLE I
MODELING PARAMETERS AND RESULTS FOR DIFFERENT AMOUNTS OF WHEEL-SLIP.

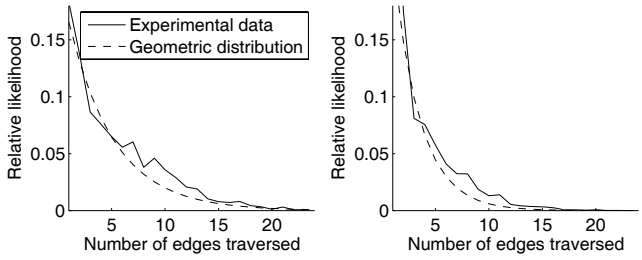

Fig. 4. The relative likelihood for successfully traversing a certain number of edges for wheel-slip of 0.1% (*left*) and 0.5% (*right*) matches a geometric distribution (superimposed).

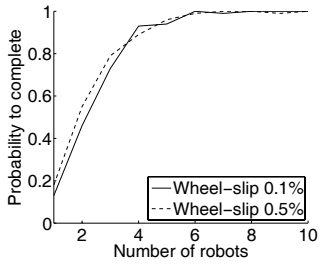

Fig. 5. Percentage of *Webots* experiments that led to complete coverage within 2h simulated time.

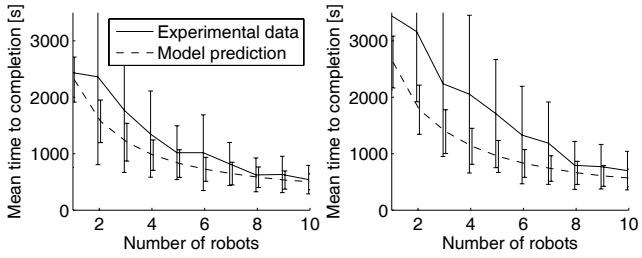

Fig. 6. Mean time to completion in a realistic simulator for team sizes of 1 to 10 and wheel-slip of 0.1% (*left*) and 0.5% (*right*). Predictions of the deterministic model for a 5x5 environment are superimposed. Error bars show the standard deviation.

and with experiments with a team of 10 robots in Tab. I (10 experiments).

For calculating the model prediction, we iterated (5)–(7) until all navigable vertices were visited at least once. Then, modeling results were scaled to seconds using the value of τ^e (Tab. I) that results from the wheel-slip programmed in *Webots*. We also calculate the average number of vertices visited by a single robot before achieving complete coverage in Tab. I.

Despite our policy of starting over on failure, not all of the experiments completed in less than two hours simulated time (two hours are three times as large as the average time for a single robot). The success rate of our experiments is shown in Fig. 5, and Fig. 6 shows the average time to completion of the successful experiments.

Finally, we measure the average number of covered blades over 100 experiments in *Webots* (including those that did not complete). Results for teams of 1 and 10 robots (lower and upper curve, respectively) for wheel-slip of 0.1% and 0.5% are shown in Fig. 7, *left*, and *right*. These results are compared with predictions from the probabilistic model (10) for parameters μ from Table I.

V. DISCUSSION

As expected, varying amounts of wheel-slip in simulation indeed led to a constant probability for violating the completeness properties of the deliberative algorithm. We conjecture that physical properties of a real robot system such as sensor and actuator noise, or reliability of another subsystem (for instance a detector to detect mines as in [7]) can be parameterized in a similar way.

We observe that the prediction error of the deterministic model becomes larger the less the individual robots are ideal. This can be seen when comparing Fig. 6 *left* and *right*, which shows a much higher time to completion for robots with higher wheel-slip than model prediction, even though the larger value of τ^e for higher wheel-slip is taken into account. Another shortcoming of the deterministic model is that it does not take into account complete failure. Although it seems that the relative divergence between model prediction and realistic simulation decreases for larger team sizes (the data for Fig. 6 is available at [15]), this is rather an artifact of the high success rate of experiments with larger team sizes (Fig. 5).

We also observe that the variance of the performance is much higher in realistic simulation than predicted by the deterministic model, and increases with the wheel-slip. This can be explained by the fact that the robots behave less

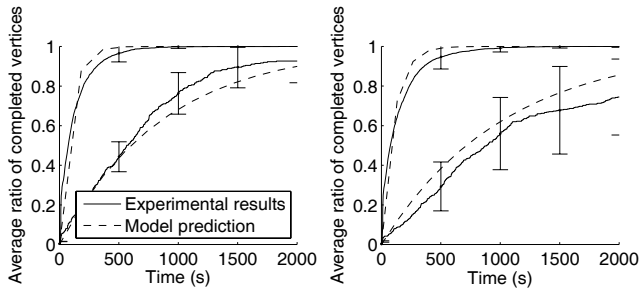

Fig. 7. Average ratio of covered blades for teams of 1 and 10 robots (coverage progress with 10 robots is faster) and wheel-slip of 0.1% (*left*) and 0.5% (*right*). Prediction of the probabilistic model is superimposed. Error bars depict standard deviation.

faithfully to their deliberative control scheme with increasing amounts of noise. A similar observation has also been made in [7] for a probabilistic search task.

Looking at Fig. 5 we observe that the probability for small team sizes to fail completely is systematically *lower* for higher amounts of wheel-slip. This is in line with the observation that noise eventually increases the robustness of reactive navigation behaviors that could otherwise get stuck in local minima.

Even though the probabilistic model (10) is much simpler than those in [5] that model the behavior of the robots more precisely, it is able to capture well the dynamics of different slip-noise and team sizes. Unlike the deterministic model, which provides the average time to completion for various drop-off locations for those experiments that do not fail completely, the probabilistic model predicts the probability for completion at a given time, and thus combines the metrics from Fig. 5 (probability) and Fig. 6 (average time) into a single expression. We remark, that the model requires no free parameters but μ and τ^e, which were shown to be a function of the individual robot's capabilities and can be easily measured.

We conclude the discussion with a comment on the experimental methods used: in order to assess (swarm) robotic problems probabilistically, large numbers of experiments are necessary to validate modeling assumptions. Here, realistic simulation showed to be a useful tool to gather experimental data in quantities that are infeasible or too costly to gather using real robot experiments.

VI. Conclusion and future work

Although this case study is very specific, it has certain properties that might well apply to other coverage scenarios or for larger robotic platforms. By treating the boundary coverage problem of our case study as a graph coverage problem, it becomes irrelevant, whether the vertices of a graph need to be entirely covered, or otherwise processed. Rather, vertex and edge relations need to be established online, a process which is prone to failure due to sensor and actuator noise, and independent from previous states of the robot. Reactive behaviors and physical interferences are critical factors that might lead to possible failure of otherwise provably complete algorithms. If the navigation error of those behaviors is not negligible (which is rarely the case), purely deterministic models are unsuitable to assess the performance of a particular multi-robot coverage algorithm.

Combining deliberative and reactive algorithms allows us to explicitly take into account the reliability of an individual platform into the modeling and design process. Analysis of the deliberative part then provides the upper bound for performance that could be achieved under perfect conditions, which *gradually* decreases under influence of noise to the lowest performance bound, calculated by probabilistic analysis.

In this paper, we assume that the probability to fail is the same for all edges and all robots, and thus summarize the reliability by a single parameter. In reality, some edges are more difficult to navigate than others, and navigation skills of robots might differ as well. In our scenario this is imposed by the geometry of the blades, which require different behaviors for traversing an edge, whereas the robotic system is homogeneous. In other scenarios and general cellular decompositions, the probability of successful edge traversal might be given by the terrain, robot capabilities, lighting conditions, or geometric constraints of the environment to name a few. Here, finding minimal spanning trees that maximize the probability to completion poses an interesting research problem, which could be investigated with the modeling framework proposed in this paper.

Acknowledgments

The authors would like to thank Clément Hongler and Julien Nembrini for their help with graph formalism.

References

[1] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, Canada, August 2005, pp. 3852–3857.

[2] N. Agmon, N. Hazon, and G. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Orlando, FL, USA, May 2006, pp. 1698–1703.

[3] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–378, 2005.

[4] H. Choset, "Coverage for robotics—a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[5] N. Correll and A. Martinoli, "Collective inspection of regular structures using a swarm of miniature robots," in *Proc. of the Int. Symp. on Experimental Robotics (ISER)*. Singapore: Springer Tracts for Advanced Robotics (STAR), Vol. 21, June 2006, pp. 375–385.

[6] Z. Butler, A. Rizzi, and R. Hollis, "Complete distributed coverage of rectilinear environments," in *Proc. of the Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Boston, MA, USA, 2001.

[7] E. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *Int. J. of Robotics Research*, vol. 22, no. 7–8, pp. 441–466, 2003.

[8] K. Easton and J. Burdick, "A coverage algorithm for multi-robot boundary inspection," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005, pp. 727–734.

[9] N. Correll, S. Rutishauser, and A. Martinoli, "Comparing coordination schemes for miniature robotic swarms: A case study in boundary coverage of regular structures," in *Proc. of the Int. Symp. on Experimental Robotics (ISER)*, Rio de Janeiro, Brazil, July 2006.

[10] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1–4, pp. 77–98, 2001.

[11] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[12] N. Correll, C. Cianci, X. Raemy, and A. Martinoli, "Self-Organized Embedded Sensor/Actuator Networks for "smart" Turbines," in *IROS 2006 Workshop: Network Robot System: Toward intelligent robotic systems integrated with environments*, Beijing, China, October 2006.

[13] N. Correll, G. Sempo, Y. L. de Meneses, J. Halloy, J.-L. Deneubourg, and A. Martinoli, "SwisTrack: A tracking tool for multi-unit robotic and biological research," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, Oct. 2006, pp. 2185–2191.

[14] G. Caprari and R. Siegwart, "Mobile micro-robots ready to use: Alice," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, Canada, August 2005, pp. 3295–3300.

[15] (2006) Supplementary material. [Online]. Available: http://swis.epfl.ch/research/coverage