Coverage Under Dead Reckoning Errors: A Hybrid Approach

Victor Shafran, Gal A. Kaminka, Sarit Kraus
Bar Ilan University
{shafrav,galk,sarit}@cs.biu.ac.il
Alcherio Martinoli
Swiss Federal Institute of Technology Lausanne
alcherio.martinoli@epfl.ch

Abstract

Coverage is a task, where a robot is to move about a given a target area until every point in it is visited. Many efficient coverage algorithms cannot be used in practice, because they assume accurate movements by the robot; unfortunately, real robots have navigational errors. A standard costly solution is to utilize a robot that continuously localizes, so as to make course corrections. In this work we present TRIM SAIL, a novel hybrid coverage algorithm that takes as input an exact-movement coverage algorithm, and a maximal dead-reckoning error bound. It optimizes use of the exact-movement algorithm, so as to execute its coverage plan while minimizing movement and localization costs. TRIM SAIL guarantees complete coverage, even under dead-reckoning errors. We present several variants of TRIM SAIL and demonstrate their efficacy in experiments using data collected from real robots.

1 Introduction

Coverage [4] is a canonical robotics task, where robots are given a target work area, and move about the area until every point in the area is covered by a coverage tool associated with each robot. This tool is assumed to be the robots' sensors or specific actuator. There exist a number of elegant and efficient algorithms for single- and multi-robot coverage, that all assume accurate and exact movements by the robot. Among these we include essentially all grid-based and cell-decomposition methods, that divide the target area into smaller cells. [14; 10; 5; 9; 8]. These algorithms output a coverage plan, which—if followed without movement errors—results in complete coverage of the work area.

Unfortunately, real robots have navigation errors—called *dead reckoning errors* [2], which prohibit the direct use of exact-movement algorithms. The problem is that accumulating position errors cause the robot to drift away from its planned trajectory. There are several task-independent approaches to tackling dead-reckoning errors: Calibration or mechanical means [2]; compensation by using relative locations of multiple robots [11]; or using a hybrid system which executes the exact-movement algorithm's coverage plan while continuously executing localization procedures (e.g.,[12; 7; 3; 13]) to correct the motion errors. Coverage presents a unique challenge and opportunity related to dead-reckoning, which is not addressed by task-independent methods. On one hand, coverage requires more accurate movements; unlike other navigation tasks, when a robot is to *cover* some area between A and B, each point in its trajectory must be covered. On the other hand, if the coverage tool is sufficiently large, then some motion errors can be ignored, as long as the points on the trajectories are within the area of the coverage tool.

We present a novel hybrid coverage algorithm, called TRIM SAIL. TRIM SAIL takes as input an exact-movement algorithm, the coverage tool size, and a maximal dead-reckoning error bound. It optimizes use of the exact-movement algorithm, so as to execute its coverage plan while minimizing localization checks and corrections, i.e., minimizing movement and localization costs (e.g., in terms of time and battery). Given the error bound, TRIM SAIL guarantees complete coverage, even under dead-reckoning errors. We present several variants of TRIM SAIL, including a worst-case variant, and average-case heuristics to reduce costs.

To evaluate TRIM SAIL, we experiment using data collected from real robots. We show that the analytical predictions for execution costs match the actual performance of the robot. We additionally show that all versions of TRIM SAIL outperform a task-independent hybrid approach, in which localizations are continuously performed to correct dead-reckoning errors. Finally, we show that TRIM SAIL's performance is not sensitive to cost estimates—thus even if it uses incorrect estimates as to the movement and localization costs, it will still perform well in practice.

2 Related Work

Early investigations of dead reckoning explored mechanical methods that reduce errors, a-priori by mounting additional specialized hardware and calibration of the robot to reduce systematic odometry errors [2]. However, dead-reckoning errors cannot be completely eliminated. There are non-systematic errors that are caused by environmental uncertainties, e.g., wheel slippage.

Increasingly, probabilistic methods [12; 7] are used to carry out the process of fusing information from sensors, over time, to reduce the localization errors (which otherwise accumulate with movement). These technique successfully reduce odometry error by comparing the data obtained from the sensors in a different point of time, taking into account the movements of the robot and the noise in the readings. They also utilize absolute location information (e.g., from GPS), if available.

In general, such methods require significant resources, and may also interfere with the robot's operation. For instance, in the RoboCup AIBO soccer league, the robots have to physically stop tracking the ball and the opponents, in order to free the camera to identify landmarks for localization. Our work thus focuses on optimizing the use of localization procedures. In particular, our work attempts to schedule localization requests during coverage tasks, so as to reduce costs.

An important motivation for our work is the prevalence of exact-motion coverage algorithms that are highly efficient, yet assume no dead reckoning errors. Choset [4] provides a survey of coverage algorithms. The Boustrophedon coverage algorithm is an efficient method, which relies on perfect localization [5; 9]. Spanning Tree Coverage (STC) [8] is another good example. STC-based algorithms divide the working area into cells of size equal to the robot tool, and build a Hamiltonian cycle that goes through all cells. While STC-based algorithms are efficient and easy to implement, they assume zero dead-reckoning errors, and fail in robots that have restricted capabilities [6].

Simultaneous Localization and Mapping [13] is a related task in which robots are required to map an unknown area, while also overcoming localization errors. The process requires making fusing sensory readings over time, and this puts additional constraints on the movements of the robots, which are not present in coverage. The techniques presented here do not target mapping.

3 Dead-Reckoning in Coverage

We restrict ourselves to *offline complete coverage*, where a map of the work area W, of size $M \times M$, is given, and the algorithms seek to guarantee that a robot visits every point in W. We focus on grid tessellation of the work-area, though in principle the techniques can be extended to other regular tessellation as well.

The robot's tool size is $D \times D$. Thus, when placed at a point p in the work-area, the robot covers a square of size $D \times D$, whose center is at p. The robot is assumed to be omnidirectional, or alternatively, be capable of moving forward and turning in place. We are given the angle α, which is the maximal deviation due to motion error (either left or right of the direction of the movement) as the robot moves in a straight line of a unit distance. The robot has a cost associated with a distance it travels, denoted by C_{drive} for each unit distance. This cost abstracts real-world cost components, such as execution time, battery usage, etc. Table 1 summarizes the notation used in this work.

Now, suppose we have an exact-motion coverage algorithm, denoted Alg_{exact}. This algorithm takes W and D as an input and computes *a coverage plan*—an ordered sequence of movements and heading changes (turns), which take the robot through cells, to completely cover W. Denote by $dist_1$ the distance the robot travels in order to perform this task. Then, the total cost of this coverage task would be equal to $C_{Alg_{exact}} = C_{drive} \cdot dist_1$. If D grows, the robot cover more area in each one of the steps. As a result, the robot needs to travel less to cover the environment, under the assumptions that its movements are accurate.

Notation	Definition
$M \times M$	The size of the work area W
$D \times D$	The size of the tool coverage
α	The dead reckoning error bound
Alg_{exact}	The exact-motion coverage algorithm
C_{drive}	The cost of drive
C_{loc}	The cost of one active localization
C_{total}	The total cost of the algorithm
q	The maximal localization precision error

Table 1: Notations used in this work.

However, dead-reckoning errors interfere in executing the coverage-plan. A robot blindly following the sequence of moves may not go through the intended cells, because dead-reckoning errors will cause its actual course to deviate.

Thus to execute the coverage plan, the robot must use localization procedures to assert its position on the intended trajectory, and to make corrections if necessary. We refer to this process as *localization*. We abstract away from the actual method of localization, and consider only the cost of this operation—in terms of time and battery power—which is denoted C_{loc}. In addition, localization has only a limited precision, bounded by $q \ll D$. If robot is localized at some position p, all we know is that robot stays in a square of size $q \times q$ that is centered at p.

The number of localizations made during coverage is denoted by N. When the robot deviates, it accumulates the additional travel distance. This accumulated distance (which includes course corrections) is denoted by $dist_2$. Then, the total cost of the algorithm is given by:

$$C_{total} = C_{drive} \cdot dist_2 + C_{loc} \cdot N \tag{1}$$

To minimize this total cost (Eq. 1), the robot must carefully balance its use of localization. When such localization checks are relatively expensive (e.g., in the RoboCup AIBO league, where robots must stop tracking the ball in order to localize), increasing the number of localization checks (N) significantly increases overall costs. On the other hand, reducing N too much requires larger corrections after each localization, and thus increases $dist_2$, the travel distance including deviations and their correction. We do this by considering the error bound α, and its relation to N.

Assuming an omnidirectional robot, we address movement in straight lines in arbitrary headings[1]. Without loss of generality, suppose that the path of the robot is in the direction of the x-axis. The ideal robot, without dead reckoning errors, will simply move in a straight line along the x-axis. A realistic robot will diverge from the straight line, with the accumulating dead-reckoning errors accelerating its departure from the x-axis.

Note, however, that localizations—and subsequent corrections—are not *constantly* required, i.e., are only required at some key locations. Suppose the size of each cell in the grid is d, $0 \le d \le D - q$. Then the straight line that Alg_{exact} generates goes through a number of $d \times d$-sized

[1]This is equivalent to assuming error-less turns in a robot that can move forward and turn in place. The relaxation of this assumption is straightforward, e.g., by requiring the robot to localize (and correct its position) with every turn.

cells. But because its coverage area $D \times D$ is actually greater than $d \times d$, it can in fact allow some deviation from the intended course. For instance, suppose the robot is to cover cells of size $d \times d$ ($d = \frac{D}{2}$). The robot can deviate by $\frac{D}{4}$ along the y-axis and still cover the cells (Figure 1).

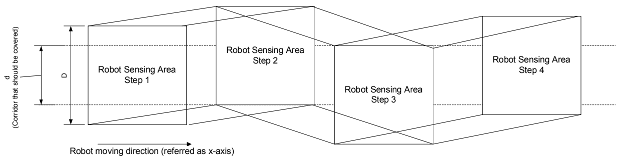

Figure 1: Example of robot motion which covers all cells, while still deviating.

This example presents an opportunity. We can control the value of d (the size of the grid used by an exact-motion coverage algorithm Alg_{exact}), such that it optimizes the use of localizations to minimize total cost. A hybrid algorithm would schedule localization actions (and their corrections) for Alg_{exact}'s coverage plan, augmenting it by periodic localization actions (and subsequent corrections, as necessary), and resulting in a complete coverage, at a minimal cost.

4 A Hybrid Coverage Algorithm

In this section we present an algorithm that utilizes a given grid-cell size parameter d, to provide complete coverage under dead-reckoning, using localizations only when necessary.

The TRIM SAILalgorithm (Algorithm 1) takes as input the exact-motion coverage algorithm Alg_{exact}; the grid-size parameter d; the robot coverage tool size D; the work area W; and α, the maximal dead-reckoning error bound (this assumes the left and right error bounds are equal; this assumption is relaxed in the experiments). It executes Alg_{exact} to create a coverage plan, and then executes the coverage plan while interleaving localization and course-corrections actions, as necessary. This results in movements as in Figure 1.

Algorithm 1 TRIM SAIL ($W, d, D, l, \alpha, Alg_{exact}$)

```
 1: $CP \leftarrow Alg_{exact}(W, d)$ {Exact-motion coverage plan}
 2: for all Plan step $stp \in CP$ (in order) do
 3:    if $stp$ is a turn or heading change then
 4:       execute $stp$ (and localize until pose is correct).
 5:    else {$stp$ is a corridor}
 6:       while corridor $Sq$ is not covered do
 7:          $(x, y, \phi) \leftarrow Localize()$
 8:          if $|Sq \bigcap Sq_{robot}| = d \times d$ then
 9:             $(r, \delta) \leftarrow$ CALCULATE$(d, D, \alpha, x, y, \phi)$
10:             Change heading by angle $\delta$
11:             Set robot to travel distance of $r$.
12:          else back-track until $|Sq \bigcap Sq_{robot}| = d \times d$
```

The algorithm first calls on Alg_{exact} to receive a coverage-plan, which assumes no dead-reckoning errors (line 1). This coverage plan is an ordered sequence of *turn* (*heading change* for omnidirectional robots) and *corridor* steps, defined as forward movement of some length. For each plan step, TRIM SAIL executes necessary localizations. Turns are executed in lines 3–4). For corridor steps, it interleaves calls to the localization action LOCALIZE() (line 7) with short movements (lines 10–11), whose angle and distance are computed in CALCULATE(), discussed below. TRIM SAIL continues this interleaved execution until the corridor is completely covered.

The robot pose (in the 2D area) is defined by three parameters (x, y, ϕ), which can be read by calling LOCALIZE(). x, y define the robot position, while ϕ defines the robot yaw (heading). We assume LOCALIZE() returns localization information with a precision defined by q.

The interleaving condition (line 8) checks whether the robot is still covering the corridor, or has possibly moved outside of it. The area that the robot currently covers is denoted by Sq_{robot}, and the corridor (of width d) is denoted by Sq, $|Sq|$ denoting the size of the area. If $|Sq \bigcap Sq_{robot}| = d \times d$ then the robot continues to cover the defined corridor. If $|Sq \bigcap Sq_{robot}| < d \times d$ then the robot deviation is too big and there is some portion of the corridor which is not currently covered. In this case, the robot needs to back-track to its previous location to re-cover the corridor (line 12).

CALCULATE (Algorithm 2) calculates the maximum distance r and heading-change δ the robot can travel until the next localization is required, under the assumption of the maximal error bound α. Using CALCULATE ensures that $|Sq \bigcap Sq_{robot}| = d \times d$ is always true, and line 12 in Algorithm 1 is never reached. However, line 12 will be used when α is heuristically estimated (Section 5). Theorem 4.1 asserts the correctness and completeness of TRIM SAIL.

Algorithm 2 CALCULATE ($d, D, l, \alpha, x, y, \phi$)

```
1: $m \leftarrow \cos 2\alpha(|y| + 0.5(D - l - d)) + 0.5(D - d) - |y|$
2: $n \leftarrow \sin 2\alpha(|y| + 0.5(D - l - d))$
3: $\theta \leftarrow \tan^{-1}(\frac{m}{n})$
4: $\delta \leftarrow \frac{\pi}{2} + \phi - \theta - \alpha$, but $\delta \leftarrow -\delta$ if $y < 0$.
5: $r \leftarrow \frac{|y| + 0.5(D - l - d)}{\cos\theta}$
6: return $r, \delta$
```

Theorem 4.1 *If $|Sq \bigcap Sq_{robot}| = d \times d$ holds at the initial position of the robot, then Algorithm 1 achieves complete coverage of the environment.*

Proof Not shown for lack of space.

The following corollary is used in Section 5. It is used in alternative methods for determining d, which affects the cost of the coverage.

Corollary 4.2 *For a distance x planned by Alg_{exact}, a robot using Algorithm 1 travels the distance $r \le \frac{x}{\cos 2\alpha}$.*

5 Reducing Localization Cost

Some of the parameters to TRIM SAIL can be arbitrarily set (d, provided to Alg_{exact}, and the error bound α). Larger values of d will result in smaller sequences of moves, but require more frequent localizations (N increases). Smaller d values allow for less frequent localizations (smaller N) but increase the correction distance. We first analytically determine the optimal value d_{min} for d, based on the maximal dead-reckoning error α, defined earlier. We then discuss estimating an average-case d, which would work well in practice.

Choosing d: Worst Case Analysis. Since the size of the map is $M \times M$, the number of cells of size $d \times d$ is $\frac{M^2}{d^2}$. Under assumption of no errors, a robot travels distance d for each cell; the total distance the robot travels is therefore $\frac{M^2}{d}$. Based on Corollary 4.2, using TRIM SAIL to overcome errors, we can conclude that the total distance including corrections is bounded by $\frac{M^2}{d \cdot \cos 2\alpha}$. Also, the total number of localizations is bounded by $\frac{M^2 \cdot \sin 2\alpha}{d \cdot (D-d-l) \cdot \cos 2\alpha}$.

Denote $D' = D - q$. We extend Equation 1 and write down the expression for the total cost of the robot's work:

$$C_{total} = C_{drive} \cdot \frac{M^2}{d \cdot \cos 2\alpha} + C_{loc} \cdot \frac{M^2 \cdot \sin 2\alpha}{d \cdot (D' - d) \cdot \cos 2\alpha} \quad (2)$$

Equation 2 is a function of d, which provides an upper bound on the cost of the coverage under dead reckoning errors. To determine an optimal d, we find d values (in the interval $[0.D - q]$) that minimize this function. Note we used a worst case α to find d_{min} value. Because it relies on a worst-case analysis, this variant of TRIM SAIL never makes corrections, but may be more expensive than a riskier variant.

Using a Heuristic α Estimate. Observing the dead-reckoning errors of real robots, we find that most of the errors are much smaller than the worst case robot error α. Thus, we can use smaller values of the α in the TRIM SAIL algorithm (and Equation 2), to reduce the number of localizations. However, this risks greater travel costs, as corrections might be required. When the actual error is larger then the α value used, the robot will need to back-track to the point where its deviation was less or equal to the one allowed by the current d_{min} and α values (Line 12 in Algorithm 1). Thus the selection of a smaller α value must be carefully balanced against the cost incurred for corrections.

We estimate α using error data measured on a real robot. We propose (and empirically compare in Section 6) three heuristics, all based on analysis of the robot errors. Given an estimated α, we utilize the analysis for d_{min} value (Eq. 2):

—*Simple Symmetric Heuristic.* Use the mean of the distribution, ignoring the error sign (errors left of heading have a positive sign, others negative). This mean value is used as α.

—*Absolute-Value Symmetric Heuristic.* Estimate the mean from all errors, while ignoring the sign of the error.

—*Non-Symmetric Heuristic.* Collect the errors of the left and right sides separately; estimate their means separately.

6 Experiments

In this section we complement the analysis from previous sections with experiments with data from real robots. The experiment settings are described in Section 6.1. The first experiment (Section 6.2) compares the data obtained from real robot with the analytic estimates. Then, we compare the performance of the TRIM SAIL coverage algorithm—and the different heuristic estimates for α—with a naïve hybrid, which uses localization continuously (Section 6.3). Finally, we conduct sensitivity analysis to examine the robustness of the techniques to inaccuracies in cost estimates.

6.1 Experiment Settings

In order to evaluate the techniques described above, we obtained error data from a Friendly Robotics RV-400 robot, and used it to simulate the robot's movements across the hundreds of robot runs used in the experiments below. To limit reliance on the choice of the exact-motion coverage algorithm Alg_{exact}, we chose to use a corridor environment, in which all algorithms behave similarly. The robot and coverage algorithm settings are described below.

Robot settings. The RV-400 is a commercial vacuum-cleaning robot, which we fitted with our own control software (Figure 2). The RV-400 runs its own coverage software, but this software was disabled in these experiments. Instead, we run our own coverage algorithms.

Figure 2: An RV-400 robot, used in experiments.

To generate a data set of dead-reckoning errors, the RV-400 robot was commanded to move in a straight line, for a distance of 40cm. This was repeated 50 times, resulting a data set of 50 measurements. For each movement, we measured the error in the robot position at the end of the movement, and calculated the resulting error in heading (angle). This data set forms the basis for the motion error models that we use in this section.

Evaluating the techniques presented above requires measuring a large number of configurations, multiple times. For instance, to evaluate the upper bound computed based on Equation 2, we vary d in the range $[0, D - q]$, and repeat each setting 50 times. We additionally vary the heuristic technique used with Alg_{exact}. This would have made for an impractical number of runs with the physical robots. We therefore chose to conduct controlled experiments by simulating the movements of the robot, using the motion errors described above. With each simulated forward movement (each step) required by the controlling algorithm (TRIM SAIL, Alg_{exact}, etc.), we randomly picked one of the error values and moved the robot under the influence of this error. The simulated robot's movements accurately simulate its movements in our lab.

We use 40cm as the basic distance unit in all experiments, and in reporting all results. The real sensor range D was set to 2 meters (5 40cm units). Using the collected errors, we found that the maximal robot deviation α_{max} is bounded by $15.6°$. All experiment results are averages over 50 trials.

Coverage algorithm settings. For simulation purposes we set the environment area to be equal to $400m^2$ (2500 tiles, 40cm each side). Since the robot's sensors have a range of 2 meters, this corresponds to a corridor of 200m by 2m (500 by 5 of the 40cm steps). The use of a corridor was motivated by two factors: First, all coverage algorithms behave similarly (if not identically) in this environment, and thus the results would not depend on our choice of Alg_{exact}. Second, as TRIM SAIL's localization in turns is the same as any other exact-motion algorithm, this environment highlights TRIM SAIL's differences with existing work. Unless otherwise noted, the different costs were set with a 1:5 ratio (i.e., $C_{drive} = 100$ and $C_{loc} = 500$).

6.2 Calculating d: The Basic Technique

We first evaluate the upper bound in Eq. 2 with real-world data. We compare the cost of using TRIM SAIL (Algorithm 1), with the values obtained from Eq. 2. We vary the virtual sensor size d. This will ensure that the minimum d_{min} computed based on Eq. 2 corresponds to the minimum in the real runs. We set d to 1, 2, 2.5, 3, 3.16 (the d_{min} value, computed based on $\alpha = 15.6°$ in the data), 3.5, 4, and 4.5 40cm steps. For each one of these 'virtual' grid sizes, we run a coverage algorithm for 50 times using the error data we obtained from the real robot.

Figure 3 presents the data obtained in these experiments. This figure compares the cost function of Algorithm 1 run in our simulation with the cost obtained from Eq. 2. It shows that indeed the real cost is bounded by the results from Eq. 2, by 14% in all the measured points. The qualitative behavior of both functions is identical. For both, $d = 3.16$ is a the minimum.

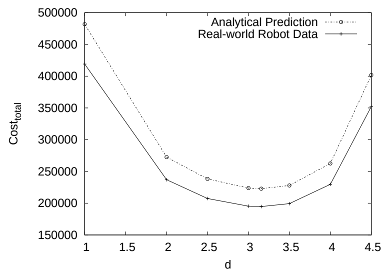

Figure 3: Comparison of running Algorithm 1 with real-world data (averaged over 50 trials), with the predicted cost obtained from Eq. 2.

6.3 Comparing Complete Coverage Algorithms

To establish a baseline for the experiments, we first run Alg_{exact}, as is, to measure its cost and coverage success. Because there are no localizations, Alg_{exact} never turns or travels to correct its location. However, its coverage percentage is poor; in the different trials, Alg_{exact} coverage percentage ran from 13.5% to 73% of the area, with a mean of 43.25%.

These results demonstrate the impact of violating the perfect dead-reckoning assumptions of many exact-motion coverage algorithms. Here, a provably-complete algorithm fails—by a significant margin—to provide complete coverage because its motion is erroneous. Many elegant exact-motion solutions to the coverage problems would suffer from similar problems. Direct comparison of TRIM SAIL to Alg_{exact} therefore does not make sense: Alg_{exact} would fail to provide complete coverage, which TRIM SAIL provides.

TRIM SAIL hybridizes exact-motion coverage algorithms, modifying their use in real-world settings, to maintain their proven properties of efficiency, robustness, etc. *while guaranteeing 100% (complete) coverage*. However, a more direct approach is possible in principle, where an exact-motion algorithm would simply be used together with continuous (repeating) localization. For instance, if landmarks are always sensed by the robot, then the robot can—in principle at least—run localization procedures without pause, resulting in continuous error corrections, and complete coverage.

We therefore turn to empirically evaluate TRIM SAIL and its heuristic variants (Section 5), against a naive use of an exact-motion algorithm with persistent localization. We compare the following techniques: Alg_{loc}, which is Alg_{exact} used with persistent localization (to create the best possible Alg_{loc}, we assume perfect localization); TS_{max} is the worst-case TRIM SAIL using the maximal heading error bound α_{max}; and TS_{simple}, TS_{abs}, TS_{ns} are TRIM SAIL variants using the simple-symmetric, absolute-value symmetric, and non-symmetric heuristics. We remind the reader that these heuristic variants attempt to reduce the number of localizations, at the risk of added travel distance for corrections.

The three heuristic methods TS_{simple}, TS_{abs}, and TS_{ns} all rely on estimating the distribution(s) underlying the error measurements. To do this, we used standard distribution-fitting procedures. We found that the results are best fitted by Pearson's Type 5 distributions, also known as *Pearson5* [1]. The distribution fit was done separately for each heuristic. The fitted mean (in the case of symmetric heuristics) or means (non-symmetric heuristic) were taken as the α value(s) used in the algorithms. For instance, for the simple symmetric heuristic, the fitted distribution had a mean of $\alpha_{simple} = 1.4703°$.

The results of the comparison appear in Table 2. Each row corresponds to a single algorithm, and the values in it are averaged over 50 trials. We use horizontal lines to distinguish the analytically-motivated algorithms Alg_{loc} and TS_{max} from the heuristic-based algorithms TS_{simple}, TS_{abs}, and TS_{ns}. The columns (left to right) provide the total distance traveled (in units of 40cm steps), the number of localization actions, and the distance/localization ratio. The final column indicates the total cost resulting from using the algorithm in question. Table 2 leads to several conclusions, explored below.

Name	Distance	Number of Localizations	Dist-Loc Ratio	Total Cost
Alg_{loc}	790.35	251	3.14	204544.98
TS_{max}	792.15	231.00	3.43	194715.00
TS_{simple}	1418.09	21.04	67.4	152329.00
TS_{abs}	973.28	33.12	29.39	**113888.00**
TS_{ns}	977.25	34.57	28.27	**115010.41**

Table 2: A Comparison of coverage results by different algorithms. All algorithms resulted in 100% coverage. Two best costs are in bold. Results averaged over 50 trials.

First, we see that under the cost ratio defined (100:500), even the worst-performing variant of TRIM SAIL—TS_{max} is better than using the exact-motion algorithm Alg_{exact} with continuous localization calls (Alg_{loc}). The distance traveled by Alg_{loc} is almost the same as TS_{max}, with a much greater number of localizations. This is because Alg_{loc} makes unnecessary corrections. Because it does not consider the geometry/size of the coverage tool, it repositions even if the area is already covered. Thus TRIM SAIL indeed offers a much more effective hybridization of the original algorithm.

Second, the results reveal a qualitative significant difference between the analytical method which seeks to guarantee performance using only the maximal error bound (TS_{max}), and the heuristic methods (TS_{simple}, TS_{abs}, and TS_{ns}) which seek to minimize cost by relying on additional knowledge (here, about the distribution of heading errors). The heuristic methods significantly outperform their worst-case counterpart, demonstrating their effective utilization of the additional knowledge they have. In particular, given that all three methods relying on our fitting the error distribution to the Pearson5 distribution, we believe that this indicates that indeed this distribution type is appropriate for modeling dead-reckoning errors. To check this, we also experimented with other distribution types, and showed that Pearson5 is indeed superior. We do not provide the details here for lack of space.

Third, while the Absolute Symmetric (TS_{abs}) and Non-Symmetric (TS_{ns}) algorithms are significantly better than all others, their results are in fact non-distinguishable (two-tailed t-test results in $p = 0.32$).

6.4 Sensitivity to Cost Estimations

The distance-localization ratio of the best algorithms (Table 2) is lower than that of TS_{simple}, though higher than that of TS_{max}. The conclusion is that the results in Table 2 might be dependent on the actual cost estimates (travel cost and localization cost), which are used in TRIM SAIL. Here, we explore the sensitivity of the results to errors in the cost estimates provided to the algorithms.

Ratio⟶ Name↓	1:10 (0.1)	**1:5 (0.2) (original)**	1:1 (1)
Alg_{loc}	330035.00	204535.00	104135.00
TS_{max}	310215.00	194715.00	102315.00
TS_{abs}	**130448.00**	**113888.00**	**100640.00**
TS_{ns}	132296.12	115010.41	101181.84
TS_{simple}	162849.00	152329.00	143913.00

Table 3: A Comparison of total costs for each algorithms, under different travel-to-localization cost ratios. Best costs are in bold.

Table 3 shows the total costs for the different algorithms, when the travel-to-localization cost ratio is systematically changed from the original settings (marked, fourth column from left). First, we note that the TS_{abs}, which we found earlier to be the best, remains so under extreme changes to the cost ratio: The result holds from a cost ratio of 1:25 until a cost ratio of 1:1. Thus one conclusion is that the top performing heuristic technique is in fact extremely robust to cost estimate errors.

A second important conclusion is reached contrasting the the top two rows (Alg_{loc} and TS_{max}). We see that TRIM SAIL provides superior performance to that of the other hybrid approach, *in all cost ratios*. This again demonstrates the efficacy of the methods we presented in this paper.

7 Conclusions

In this paper we presented TRIM SAIL, a hybrid coverage algorithm (and associated heuristics, geometric optimizations) for real-world settings. TRIM SAIL takes an exact-motion coverage algorithm, which assumes no dead-reckoning errors, and uses it to guide angled movements that guarantee complete coverage of the target work area, while minimizing the use of localization to that strictly necessary. We presented an analytical worst-case version of TRIM SAIL, and three heuristics which further reduce total coverage costs. We then reported on extensive experiments with TRIM SAIL, using data collected from the RV-400 robot. The experiments demonstrated that (1) the analytical methods accurately predict an upper bound for total costs, and minimum cost, given robot error bounds and coverage range; (2) the heuristic methods outperform the analytical methods in the cost ratio chosen; (3) TRIM SAIL variants are not sensitive to errors in cost estimates; and (4) that the TRIM SAIL algorithm *always* outperforms naive coverage hybridization, where the exact-motion algorithm is simply coupled with continuous localization. In the future, we hope to explore new heuristic directions which take more risks in terms of completeness of coverage, but provide reduced costs.

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1964.

[2] J. Borenstein., H. Everett, and L. Feng. *Navigating Mobile Robots: Sensors and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.

[3] A. Burguera, G. Oliver, and J. Tardos. Robust scan matching localization using ultrasonic range finders. In *IROS-05*, pages 1367–1372, 2005.

[4] H. Choset. Coverage for robotics - A survey of recent results. 31(1–4):113–126, 2001.

[5] H. Choset and P. Pignon. Coverage path planning: The Boustrophedon decomposition. In *International Conference on Field and Service Robotics*, 1997.

[6] N. Correll and A. Martinoli. Distributed coverage: From deterministic to probabilistic models. pages 379–384, 2007.

[7] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *ICRA*, pages 1322–1328, 1999.

[8] N. Hazon and G. Kaminka. On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 2008.

[9] W. H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. volume 1, pages 27–32, 2001.

[10] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer, 1985.

[11] I. M. Rekleitis, G. Dudek, and E. E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *IJCAI97*, pages 1340–1345, 1997.

[12] S. Thrun. Finding landmarks for mobile robot navigation. In *ICRA*, pages 958–963, 1998.

[13] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[14] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *In Proceedings of International Conference on Advanced Robotics*, pages 533–538, 1993.