Coordination of Multiple Vehicles for Area Coverage Tasks

Garrett Winward Nicholas S. Flann

Abstract— **Area coverage operations such as plowing a field or mowing a lawn can be performed faster if multiple vehicles are involved. To use a team of automated vehicles safely and effectively they must be coordinated to avoid collisions and deadlock situations. Unexpected events may occur during the operation which may affect vehicles' velocities, so the coordination method must be robust with respect to these events. In this paper, a path coordination method is introduced which delays decisions about mission coordination as long as possible during mission execution so such unexpected situations are efficiently handled. The method's computation speed and solution quality are evaluated through simulation, and compared with two other methods based on common path coordination techniques.**

I. Introduction

Automated vehicles can perform many tasks that are dangerous or undesirable for human operators. When controlled correctly, they can also be more efficient and precise than human drivers. A key problem with control of autonomous vehicles is path planning. Path planning involves determining the shortest or most fuel efficient route to reach a destination, avoiding any obstacles along the way.[1].

Area coverage applications are a special case of path planning. In these applications, the goal is to cause an implement mounted on the vehicle to pass over every point inside a specified region [1]. Examples of this include lawn mowing, mine sweeping, vacuum cleaning, and agricultural applications such as plowing or harvesting a field.

Area coverage operations can be performed faster using multiple vehicles. However, the problem becomes more complex when multiple vehicles are involved, as they must then consider interactions with each other. Collisions between vehicles must be avoided. Deadlock situations also may occur when there is a cycle of vehicles which cannot proceed safely because each must wait for another vehicle to move first. These situations need to be prevented to allow all vehicles to finish the mission. Vehicles must be temporally coordinated to avoid these problems and to cooperate effectively.

To reduce the complexity of coordinating multi-robot missions, often decoupled methods are used. These methods divide the problem into two parts. In the first part, paths are created for each vehicle individually, giving no consideration to the other vehicle's paths. In the second part, vehicles are coordinated to avoid collisions and deadlocks by modifying their velocities [2]. This paper discusses the coordination aspect of the multi-vehicle area coverage problem given a set of pre-planned coverage paths.

G. Winward is with Research and Development, Autonomous Solutions, Inc. Garrett.Winward@asirobots.com

N. Flann is with the Department of Computer Science, Utah State University Nick.Flann@usu.edu

II. Problem Definition

The multi-vehicle area coverage problem may be defined as follows. Given a set of unmanned vehicles and coverage paths control those vehicles in such a way that every point inside a specified area is covered exactly once by one vehicle. Several constraints must be met in accomplishing this goal.

a) Constraint 1: Collision Avoidance: Vehicles must not collide with each other.

b) Constraint 2: Deadlock Avoidance: In our application a deadlock occurs when vehicles are arranged in such a way that no vehicle can move to a new position unless another vehicle vacates that position first, creating a cyclic wait. A simple example of this is when two vehicles come to a stop nose-to-nose, so neither can move safely without colliding with the other. Vehicles can't give up the space they occupy, so the only way to avoid deadlocks is to ensure that cyclic waits will not occur.

c) Constraint 3: Precise velocity control of vehicles throughout the mission should not be necessary to ensure safety: In area coverage operations, unexpected events can occur during runtime that require vehicles to adjust their velocities. For example, a plow may run over a particularly rough or rocky area and be forced to slow down or it will risk being damaged. In other situations, the vehicle may not always be able to move at its desired velocity due to mechanical failures or weather conditions.

It will not always be possible to anticipate such events, so the safety of the system cannot depend on vehicles being in certain places at specified times. Estimates of time taken to reach a position may be taken into account to try to formulate efficient solutions, but the safety of the system must not rely on these estimates. This means that a solution satisfying the first two constraints must contain some kind of reactive component that can adjust vehicle velocities during mission execution.

d) Constraint 4: Deviation from planned paths is not allowed: Area coverage requires vehicles to pass over every point exactly once inside a given area. If a vehicle deviates from its path inside the area to avoid collision with another vehicle, it will leave some area uncovered, while potentially covering some other area twice.

If a vehicle is performing a lawn mowing operation, swerving creates irregularities in the coverage pattern, which make it less aesthetically pleasing. Swerving behaviors are illegal in these applications and must be avoided. This means that vehicles must follow set planned paths, which in turn means that only the velocities of the vehicles may be modified to satisfy constraints 1 and 2.

Vehicles also should be coordinated so they cooperate to cover the area as efficiently as possible. Due to constraints 1, 2, and 4, at times it will be necessary for vehicles to pause to allow other vehicles to progress to a certain point before moving on. Such pauses should be minimized as much as possible in order to allow the task to be finished quickly and maximum benefit to be gained from using multiple vehicles.

III. Related Work

A. Path Planning

Several works have been published dealing with planning paths for area coverage of known spaces. [3] discusses a method which generates a spanning tree throughout the area to be covered, then creates the coverage path by growing a boundary around the spanning tree. [4] uses a neural network to create a gradient field throughout the coverage area, drawing vehicle paths to follow the gradient uphill until coverage is obtained.

The most commonly used coverage planning technique is known as the Boustrophedon Cellular Decomposition, introduced by [5] This method divides the area into a set of subproblems, then covers each subproblem using a series of parallel passes. A multi-vehicle adaptation of this method is used to generate coverage plans used in this paper.

Many methods exist which combine dynamic adapatation of path plans with schedule coordination to avoid collisions and deadlock situations. These methods are not discussed in this paper as they involve deviating from the originally planned paths, which in our application is undesired behavior and violates Constraint 4.

B. Path Coordination

One of the earlier works which coordinates robot manipulators by building the collision space and planning a path through it to find a schedule is [6]. This work considered problems involving only two robots.

A version of the previous algorithm is discussed in [7] that works with any number of vehicles and uses bounding boxes to represent obstacles in the collision space. It makes use of the observation made by [2] that obstacles in the collision space are cylindrical, so only the $n(n-1)/2$ graphs between each pair of robots need be represented. It also identifies subsets of intersecting robot paths which can be considered as separate coordination problems.

A decentralized version of this algorithm is introduced in [8] which works on a localized area when robots approach each other.

A Mixed Integer Non-Linear Programming method is formulated in [9] using a kinodynamic model of vehicle motion taking into account maximum speed and acceleration rates. A pair of Mixed Integer Linear Programming problems are then defined representing the upper and lower bounds of motion through each zone, which is then used to determine an optimal schedule for all robots. Vehicles must then follow that schedule exactly.

Many methods consider only previously planned paths when determining a schedule of vehicle motions. In [10] solutions are found by simulataneously searching for the new robot's path and for the coordination of motions of robots along previously planned paths. In [11] conflicts are resolved according to a predetermined absolute priority ordering of vehicles, and uses a search to determine the best priority ordering to use.

Each of these methods relies on the robots following precise velocity schedules. As mentioned earlier, in our problem domain we are concerned with robustness in the face of unexpected events which may throw vehicles off their planned schedules, so these methods are not appropriate for use in our application as they are described.

Another path coordination approach is discussed in [12]. This method is used by a reactive planner to coordinate vehicles when they get close to each other. Paths are analyzed to find intersections between vehicle paths. Checkpoints are identified at positions on the path where vehicles may safely pass by each other. Vehicles lock out the path between these checkpoints so only one vehicle at a time will enter it. This method works for solving pairwise problems, however, area coverage situations often introduce more complex situations involving three or more vehicles which are not handled by this algorithm.

IV. Approach

A. Definitions and Terms

In this paper, a task is defined as a single objective for a single vehicle to perform. A mission is a larger scale objective consisting of multiple tasks. When we refer to the mission, we mean all paths planned for all vehicles in the system.

Since each vehicle is constrained to follow a set path, the space that will be occupied by the vehicle throughout the mission may be estimated. By sweeping the outline of the vehicle and its implement along the path, a bounding volume is created that represents this space. This swept volume is called a trace [13][7].

Each vehicle's progress along its path may be represented by a single variable, which is designated as λ. The Cartesian product of two vehicles' λ values represents all possible configurations those two vehicles may be in during their mission. A graph may be formed, plotting each λ value along one of the axes. This graph is referred to as the collision space [9][6][7].

By comparing the traces of two vehicles, we places can be determined where potential collisions may occur. At any point where traces intersect, the corresponding λ values for each vehicle are recorded. These points are then plotted in the collision space as obstacles to be avoided. The pair of intervals defined by the λ values beginning and ending an obstacle are known as collision zones [9]. A path through the collision space that intersects no obstacles constitutes a collision free schedule of coordinated motions. [6] [7].

For more than two vehicles, the collision space is formed by the Cartesian product of all λ values for all involved vehicles. The dimension of the space is the same as the number of vehicles involved in the mission. Thus, the size of

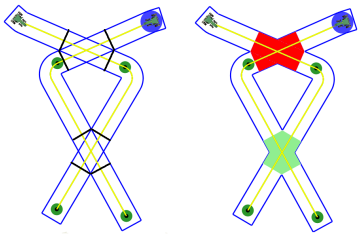

Fig. 1. This figure shows the creation of collision zones for a pair of vehicle paths. Traces are formed around the vehicles' paths. These traces are analyzed and intersecting regions are found. The red area signifies an opposing collision zone, while the light green area represents a parallel zone.

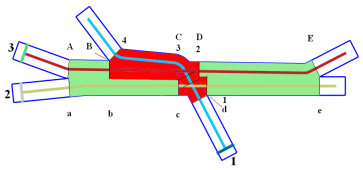

Fig. 2. This figure shows three vehicles' paths labeled with large numbers, and their resulting collision zones. The letters and numbers represent distances along each vehicles path, which are used to define the collision zones. This example has three zones: a parallel zone between vehicles 2 and 3 with intervals $[A, E][a, e]$, an opposing zone between vehicles 1 and 2 with intervals $[1, 3][c, d]$, and an opposing zone between vehicles 1 and 3 with intervals $[2, 4][B, D]$

the space, (as well as the complexity of the search problem to determine an optimal schedule) increases exponentially with the number of vehicles [7].

B. Collision Zone Creation

Many classical approaches estimate collision regions using bounding boxes for speed of calculation and ease of planning. In cases where intersecting regions are short, using bounding boxes is sufficient to allow for near-optimal schedules to be found.

Unfortunately, this is too restrictive for our problem domain. In area coverage applications, vehicles cover a field in a series of parallel passes. These passes can be very long, in some cases even longer than a mile. Representing zones as bounding boxes introduces a large amount of unnecessary waiting time when two vehicles travel over adjacent rows going the same direction. One way of handling this problem is to subdivide the passes into smaller sections. However, doing so adds obstacles to the collision space, further increasing the complexity of a search.

For our approach we will handle this problem by specifying two different types of collision zones. The two types are defined as follows. When vehicles travel in opposite directions, if both vehicles enter the zone at the same time, a deadlock must necessarily occur. This situation is designated as an *opposing collision zone*.

When vehicles travel in a roughly parallel direction, as long as enough space is given on entrance of the zone, the leading vehicle will always be able to move. Both vehicles may be in the zone at the same time. This is designated as a *parallel collision zone*.

Collision zones are built using the following offline method, after paths have been planned for all vehicles but before they begin moving. This process is illustrated in Fig. 1

Given a pair of swept volumes, we arbitrarily select one vehicle. We then step along this vehicle's path, noting each point its trace begins or finishes intersecting the other vehicle's trace. These points are shown by black lines in the first picture in Fig. 1. Intervals are identified for intersecting traces within which the heading angle at each point along one path is less than 90 degrees from the heading at the corresponding point on the other path. These intervals are identified as parallel collision zones. Intervals are also identified within which the heading angles of the two paths are greater or equal to 90 degrees. These are identified as opposing collision zones. When analyzing the vehicles' traces, special cases at inflection points where the paths transition between parallel and opposing regions must be taken into account.

Once collision zones have been created between all pairs of vehicles, all information needed to coordinate them is available. Note that because collision zones as represented only as pairs of intervals, there is need to explicitly represent the collision space or even the individual primitive graphs, as is required by other path coordination methods.

C. Graph Creation

The next step involves building a graph from the collision zones that will be used to check for cyclic waits. This process is illustrated with the three-vehicle example shown in Fig. 2.

Nodes in this graph will represent the individual intervals of each collision zone, as shown in Fig. 3. Edges are determined as follows. Each zone interval is compared with all overlapping collision zone intervals associated with the same vehicle to determine those which must be open to allow the current zone to be traversed.

If the current zone is opposing, entering it will block other vehicles' progress. We must then guarantee that the other intersecting interval will be clear before the vehicle can be allowed to enter the current interval. We represent this relationship with an edge from the current interval to the other zone's opposite interval.

In the example shown in Fig. 3, intervals $[1, 3]$ and $[2, 4]$ overlap, which means if Vehicle 1 enters interval $[1, 3]$, it will have to enter $[2, 4]$ before it can leave $[1, 3]$. If Vehicle 1 is to be allowed to enter interval $[1, 3]$, interval $[B, D]$ also must either be clear or must be able to be cleared to allow Vehicle 1 to progress, hence the edge in the graph from $[1, 3]$ to $[B, D]$.

If the current zone is parallel, entering it will not block other vehicles' progress but vehicle ordering must be maintained. No passing is allowed, so the first vehicle in must also be the first out. Because of this we need only consider the end of the interval and certain points within it where we must verify vehicle ordering.

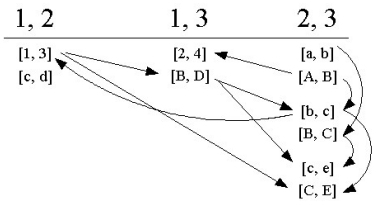

Fig. 3. This figure shows the collision zone graph created from the example in Fig. 2. The pair of numbers at the top indicate the vehicle numbers associated with the collision zones beneath them. Each interval $[X, Y]$ represents an interval in a collision zone, so each pair of intervals constitutes a complete collision zone. Each interval is a node in the graph, arrows between intervals represent edges.

Critical points for determining graph connections include the beginning of opposing zones and the end of parallel zones. The other points to be considered in parallel zones then include all places corresponding to a critical point occurrence in the zone's partner interval. In our algorithm, we accomplish this by splitting the parallel zones at these critical points before identifying the graph edges.

In the example from Fig. 2, critical points occur inside the parallel zone $[A, E][a, e]$ at points B and c, corresponding to the beginning of the two opposing zones. Splitting the parallel zone at these points creates three parallel zones, $[A, B][a, b]$, $[B, C][b, c]$, and $[C, E][c, e]$. Interval $[b, c]$ now contains the start point of interval $[c, d]$ in opposing zone $[1, 3][c, d]$ between vehicles 1 and 2, so an edge is created from $[b, c]$ to $[1, 3]$.

There is now a cycle in the graph, if all three intervals $[1, 3]$, $[B, D]$, and $[b, c]$ are occupied by a vehicle at the same time, a deadlock has occurred.

D. Online Coordination

To use collision zones to coordinate vehicle motions, The following observations are made. In an opposing zone, if one vehicle enters the zone, it must also leave the zone before the other vehicle will be allowed to enter it. Thus, any other collision zone which begins inside an opposing zone will also be entered first by that vehicle. In a parallel zone, the first vehicle into the zone must also be the first to leave it. Therefore, any collision zone whose beginning coincides with the end of the parallel zone will also be reached first by the first entering vehicle.

We use a method similar to a software semaphore to coordinate our vehicles during the mission in a way similar to what is done in [12]. In this case, collision zones are the shared resource we wish to keep vehicles from accessing at the same time. When a vehicle first enters a collision zone, it locks it. When a collision zone is locked, the other vehicle is not allowed to enter it, and must come to a stop short of the beginning of the zone.

In our algorithm, we handle parallel and opposing collision zones in different ways. Because no two vehicles may enter an opposing zone at the same time, the zone is unlocked only when the vehicle completely leaves it. Two vehicles may enter a parallel zone at the same time if they are spaced sufficiently, so a parallel zone is unlocked as soon as the vehicle has moved entirely inside of it.

This is sufficient to ensure that deadlock will not occur between any pair of vehicles. However, a deadlock may still occur due to a cyclic wait between three or more vehicles. Our method solves this problem using the graph created in the previous step. As a vehicle approaches a collision zone, the graph is checked to see if locking that zone would create a cycle. Any zones formed from the earlier splitting operation are still considered part of the original zone, and so are also checked. If no cycle is created, the zone is locked for the vehicle so no others may enter it. If a cycle would be created, the vehicle is locked out of the zone, and is commanded to slow down or come to a stop an appropriate distance from the zone's beginning. If a vehicle's path begins or ends inside another vehicle's path, the corresponding collision zones are locked on initialization. This is sufficient to ensure that no cyclic wait will occur and all vehicles will be able to finish their tasks, provided a deadlock free solution exists.

To illustrate this process, the three vehicle example shown in Fig. 2 and its resulting graph in 3 is used. Vehicle 2 first reaches point a along its path. When it does so, it enters and locks parallel collision zone $[a, b][A, B]$. Because parallel zones $[b, c][B, C]$ and $[c, e][C, E]$ were split from the initial parallel zone, they are considered part of the zone being entered and so are also locked. Once point a is passed, the first parallel zone is unlocked and Vehicle 3 is allowed in.

Next, Vehicle 1 approaches opposing zone $[1, 3][c, d]$. The graph is referred to, and no cycle of closed zones is found, so Vehicle 1 enters the zone and locks it. If Vehicle 2 approaches point c then, it will see the zone is locked and come to a stop.

Next, Vehicle 3 approaches point B and the beginning of opposing collision zones $[2, 4][B, D]$. The graph is referred to. At this point, Vehicle 1 has locked interval $[1, 3]$ and Vehicle 2 has locked interval $[b, c]$. The graph shows a cycle between intervals $[1, 3]$, $[b, c]$ and $[B, D]$, so if vehicle 3 enters interval $[B, D]$, a deadlock will occur. Vehicle 3 instead is locked out of zone $[2, 4][B, D]$ and must come to a stop. It will be allowed to proceed once Vehicle 1 passes point 4 and unlocks the collision zone.

One final provision is required to ensure that no collision may possibly occur between vehicles. Since both vehicles are allowed into a parallel zone at the same time, if the trailing vehicle travels faster than the leading vehicle, they could collide. Therefore, runtime proximity detection is needed to ensure the following vehicle maintains enough spacing.

V. Test and Results

A. Setup

In this evaluation, the path coordination method introduced in the previous section will be referred to as the Choreographer. To evaluate the Choreographer, it was compared with two other path coordination methods. The first method is based on the algorithm presented in [7]. An optimal schedule is determined by a search through the collision

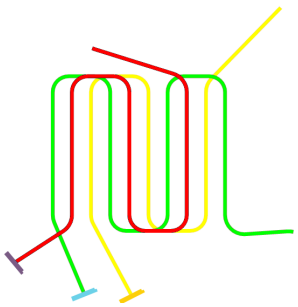

Fig. 4. This figure shows an example three vehicle coverage scenario used in testing

space as represented by the $n(n-1)/2$ collision graphs formed between each pair of vehicles.

The second method applies a priority scheme to the coordination problem such as is discussed in [11]. An absolute ordering of vehicles is determined, the collision space is formed, but instead of searching the space, vehicles earlier in the ordering are used as obstacles for later vehicles. A random restart hill climbing search is used to find a schedule.

The following modifications to these methods are used to make them applicable to our problem domain. First, parallel collision regions are divided into smaller sections. Second, we modify the methods so they do not allow collisions if vehicles deviate from their planned schedule. Each method produces as an output a schedule represented by a path through the collision space. If we note which side of the path each collision zone obstacle lies on, we know which vehicle should be allowed into that zone first. Each collision zone can then be locked accordingly on initialization. This ensures the schedule will be enforced.

These methods were tested on a set of example missions with overlapping paths. This set consisted of a number of large coverage problems taken representative of some of our actual coverage applications, as well as smaller and simpler coverage examples. Also included were several simple crossing examples like the example used in Fig. 2.

Coverage paths were generated using a multi-vehicle adaptation of the coverage planner developed by Autonomous Solutions, Inc. The algorithm used to generate coverage paths is closely related to the Boustrophedon Cellular Decomposition planning method [5]. This algorithm works on known areas where any internal obstacles are known *a priori*. Coverage paths created by this method consist of a connected series of alternating parallel passes through the area, as shown in Fig. 4. In some cases, randomized row orderings were used to add more variation to the coverage paths and simulate more complex coverage problems. All tests were performed on an Intel T2600 2.16GHz Dual Core processor running with 2 GB of RAM.

B. Computation Time

First, the Choreographer is evaluated in terms of computation speed. The growth in computation time as a function of the number of vehicles is compared between the three

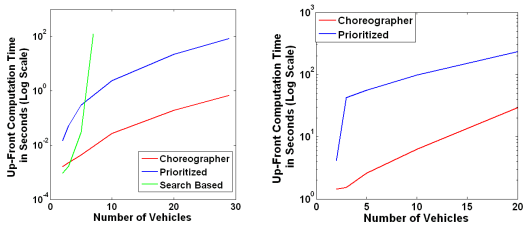

Fig. 5. This graph shows results comparing up-front computation time between the three methods. The results depicted in the first graph are from tests run on a simple scenario where all paths pass through a single common intersection point. The results shown in the second graph are from tests run on a large area coverage problem. Time is measured in seconds. Standard deviations for the simple intersection scenario at 29 vehicles were 9.71 seconds for the prioritized method and .013 seconds for the Choreographer. For the coverage scenario, at 20 vehicles the standard deviations were 38.72 seconds and 4.20 seconds, respectively. In both scenarios the standard deviation values decreased proportionally as the number of vehicles was reduced.

methods. Results are shown in Fig. 5.

As expected, the search based method took large amounts of time and was generally unusable on small problems after about 7 vehicles, and was even infeasible for large coverage problems employing just 3 vehicles. The reason is that in area coverage problems, a large number of collision regions are generated; for large problems we generally have several hundred. A large number of obstacles in the collision space greatly increases the resolution of cell divisions which greatly increases the size of the search space. Dividing long parallel regions into subsections further exacerbates this problem.

The prioritized method was usable for all sizes of coverage problems. However, the Choreographer showed a clear advantage in computation time as problem size increased. As shown in Fig. 5, in the large coverage problem the Choreographer's up front computation time was generally less than the prioritized method's by one order of magnitude; in the simple intersection example it was smaller by two orders of magnitude. This would be particularly useful in an environment in which paths are being planned dynamically, with vehicles continuously entering and leaving the system.

It must also be noted that the Choreographer exchanges some time taken up front for extra computation during mission execution. In the worst case on the largest coverage problems, a zone-entry query took approximately .3 seconds. When coordinating real autonomous vehicles this computation delay time must be taken into account, so vehicles look ahead to begin such queries early enough that they still have ample time to stop if necessary once the query is complete.

C. Solution Quality

Second, we evaluate the Choreographer's quality and robustness. Two metrics are commonly used to evaluate such problems[2]. The first is total time to completion of the mission. Essentially this boils down to the time taken for the last vehicle to finish the mission. The second metric is the summation of all the waiting times of all vehicles during the mission.

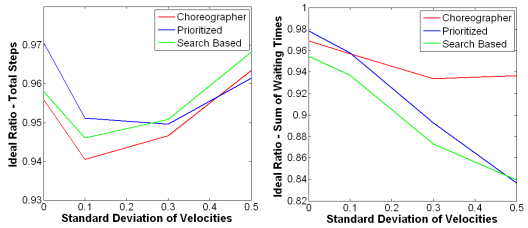

Fig. 6. This figure plots the quality of the coordination methods as a function of the standard deviation affecting the vehicle's random velocities. The results plotted in these graphs were taken from tests on a small three-vehicle coverage example. The first graph measures quality according to the Total Time Ideal Ratio metric. The second graph measures quality according to the Sum of Waiting Times Ideal Ratio metric. In both cases values close to 1 indicate more efficient path coordination.

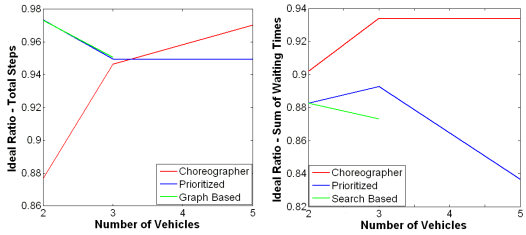

Fig. 7. This figure plots the quality of the coordination methods as a function of the number of vehicles. The results plotted in these graphs were taken from tests on a small coverage example using a standard deviation for assigning random velocities of .3. The first graph measures quality according to the Total Time Ideal Ratio metric. The second graph measures quality according to the Sum of Waiting Times Ideal Ratio metric. In both cases values close to 1 indicate more efficient path coordination.

The total time metric is generally dominated by a single vehicle; in order to minimize time it makes sense to keep the vehicle with the longest path moving, even if it requires long delays by other vehicles. The sum of waiting times metric rewards minimizing delays for all vehicles evenly. The first is a better measurement if the goal is to finish the coverage in as short a time as possible, the second if a vehicle may be retasked when it finishes its path, or if fuel consumption while idling in the field is a concern.

Many things may affect the velocities of vehicles. Mechanical errors or changes in terrain or weather may occur, and may be handled differently by different vehicles. To simulate some of these effects, we set the velocities of each vehicle in our simulation using a random normal distribution with a mean of 1, varying the standard deviation from 0 to .5.

To normalize our results across problems with different lengths of paths, we compute a ratio of the actual results against the number of steps the mission would have taken to complete if no pauses were necessary. The number of steps taken to finish the mission divided by number of steps it would have taken if no pauses were necessary will be referred to as the Total Time Ideal Ratio. The sum of the number of steps taken by all vehicles divided by the sum of the number of steps each would have taken if no waiting were necessary will be referred to as the Sum of Waiting Times Ideal Ratio.

Result distributions for these tests were massively skewed because in many cases it was possible to reach the ideal value of 1. For this reason, the significance of the results was verified with a t-test. In all cases shown, t-test comparisons on the distributions with the most dissimilar means yielded a p value of $< .00001$. Test results are shown in Fig. 6 and Fig. 7.

In terms of total time to completion, the graph search based and prioritized methods generally performed better. The discrepancy between these and our method is reduced somewhat as the variation in velocities gets larger and more vehicles are added to the problem.

The main reason for this is that in some coverage situations, if the vehicles enter the field in the wrong order, a large amount of waiting time may be necessary to ensure collision free paths. These situations become rarer as more vehicles are included in the mission, but the other methods tend to handle them better when they do happen since our method is somewhat greedy. For this reason, one avenue of future work we are pursuing is to include a one-step lookahead search when deciding whether or not to enter a collision zone, in order to detect and better handle such situations.

Despite this, our method was generally superior in terms of the sum of waiting times. This became especially true as the number of vehicles involved in the problem increased and as the variation in velocities was made greater. In these situations, the possibility for error due to deviation from the planned schedule is higher, which results in vehicles waiting for longer times because other vehicles were not there when expected.

VI. Conclusion

Area coverage operations can be performed faster if multiple vehicles are involved. To use a team of vehicles safely and effectively they must be coordinated to avoid collisions and deadlock situations. Unexpected events may occur during the operation, so the coordination method must be robust with respect to these events.

In this paper a path coordination method for use in multi-vehicle coverage operations was introduced and evaluated. We evaluated this method in terms of computation speed and solution quality through simulation, and compared our results with coordination methods based on recent coordination space planning and prioritized methods.

The method introduced in this paper is most applicable for use in uncertain environments and situations where large numbers of vehicles have heavily overlapping paths. Such situations are commonly seen in large multi-vehicle area coverage operations. This method makes it possible to efficiently coordinate a large team of vehicles performing these cooperative area coverage operations.

VII. Acknowledgments

The authors gratefully acknowledge Andrew K. Rekow at John Deere for project funding, Sarah A. Gray at Autonomous Solutions, Inc. for project management, and Daniel Coster of the Department of Mathematics and Statistics at

Utah State University for help with the statistical analysis of the solution quality measurements.

REFERENCES

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

[2] S. LaValle and S. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions of Robotics and Automation*, vol. 14, pp. 912–925, 1998.

[3] Y. Gabriely and E. Rimon, "Spanning tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 77–98, 2001.

[4] S. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Transactions on Man and Cybernetics Systems, Part B.*, vol. 34, pp. 718–724, 2004.

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

[6] P. ODonnell and T. Lozano-Perez, "Deadlock-free and collision-free coordination of two robot manipulators," *IEEE Int'l Conference on Robotics and Automation*, vol. 1, pp. 484–489, 1989.

[7] T. Simeon, L. S., and L. J., "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 42–49, 2002.

[8] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," *Proceedings of the IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, vol. 3, pp. 1213–1219, 2001.

[9] S. Akella and J. Peng, "Coordinating multiple robots with kinodynamic constraints along specified paths," *International Journal of Robotics Research*, vol. 24, pp. 295–310, 2005.

[10] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," *IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, pp. 5960–5963, 2006.

[11] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvably priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, pp. 89–99, 2002.

[12] S. Oliver, M. Saptharishi, J. Dolan, A. Trebi-Ollennu, and P. Khosla, "Multi-robot path planning by predicting structure in a dynamic environment," *Proceedings of the First IFAC Conference on Mechatronic Systems*, vol. 2, pp. 593–598, 2000.

[13] T. Simeon, S. Leroy, and J. Laumond, "A collision checker for car-like robots coordination," *IEEE Int'l Conference on Robotics and Automation*, vol. 1, pp. 46–51, 1998.