Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

A Coverage Algorithm for Multi-robot Boundary Inspection*

Kjerstin Easton (*kjerstin@caltech.edu*) and Joel Burdick (*jwb@robotics.caltech.edu*)
Division of Engineering and Applied Science
California Institute of Technology, Pasadena, CA 91125

***Abstract*— This paper introduces the multi-robot *boundary coverage problem*, wherein a group of k robots must inspect every point on the boundary of a 2-dimensional test environment. Using a simplified sensor model, this inspection problem is converted to an equivalent graph representation. In this representation, the coverage problem can be posed as the k-Rural Postman Problem (kRPP). We present a constructive heuristic which finds a solution to the kRPP, then use that solution to plan the robots' inspection routes. These routes provide complete coverage of the boundary and also balance the inspection load across the k robots. Simulations illustrate the algorithm's performance and characteristics.**

***Index Terms*— Robot coverage, multiple robots, planning algorithms.**

I. Introduction

This paper introduces the *multi-robot boundary coverage problem* and describes a graph-based methodology for planning the paths of robots that cooperate in the coverage task. In this problem (which is more fully described in Section II), a group of k robots is required to completely inspect the boundary of all 2-dimensional objects in a test environment. There are a number of practical mechanical inspection, surveillance, and security applications for such a procedure. For example, consider the problem of inspecting in-situ the surfaces of turbine blades inside a jet engine or combustor. Fig. 1 shows an idealized view of the interior of a turbine, where the cylindrical geometry of the turbine has been "flattened." Using nondestructive sensing technologies, a group of robots (which may be small mobile robots, or the distal tips of manipulators carrying the inspecting sensors) must systematically inspect the surface of each turbine blade for defects. Inspecting the freespace between the blades, as would be done in the classical coverage problem, is not of concern here. In this paper we consider a 2-dimensional version of this problem, i.e., a horizontal cross-sectional slice through this world. The extension of our method to the geometry of Fig. 1 (where the robot must inspect the entire height of the turbine blade extrusion) is relatively straightforward. Robotic security, surveillance, and "watchman's route" problems are another class of applications. Imagine, for example, that a group of robots is tasked with monitoring the walls of an art gallery, or the surfaces of a set of oil tanks. A multiple-robot approach to such problems should allow the task to be completed or repeated more quickly than with a single vehicle, while offering flexibility through redundancy in case a unit should fail[1].

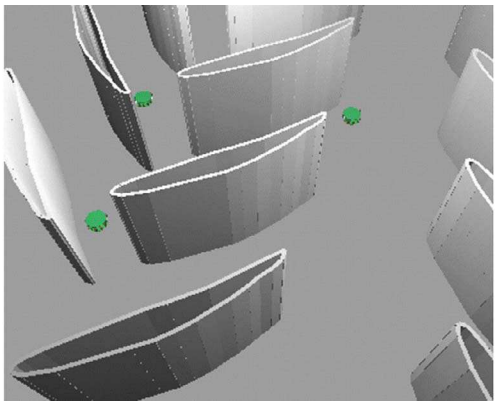

Fig. 1. Nondestructive inspection of blade surfaces inside a turbine is a motivating application of the multi-robot boundary coverage problem. The green dots represent robots engaged in sensing tasks.

In the classical coverage problem, one or more robots must "cover" the freespace of a bounded 2-dimensional workspace. Reference [1] offers a survey of single- and multiple-robot approaches to this *freespace coverage problem*. The boundary coverage problem described in this paper complements the freespace coverage problem; analogous to that problem, we seek a complete algorithm that guarantees coverage of all boundary points. Other nondeterministic approaches to a similar problem are presented in references [2] and [3]. Likewise, we are concerned with the efficiency of the robots' efforts and seek to balance the workload as evenly as possible between the cooperating robots, but our primary concern is guaranteeing completeness of coverage. We present in this paper an off-line planning procedure that meets all of these objectives.

Section II describes the boundary coverage problem and our modeling assumptions. Section III transforms the coverage problem into an equivalent graph representation, which consists of an undirected, connected graph G, which has a subset of edges E_R, in which each $e \in E_R$ represents an inspection path for a specific boundary segment. The task of routing the robots to complete the inspection task can be posed as an NP-hard graph analysis problem we call the *k-Rural Postman Problem (kRPP)*. Section IV provides a constructive heuristic to solve the kRPP, then, based on this procedure, develops a boundary coverage planning

*This work has been supported by a National Science Foundation fellowship, a grant from NASA Glenn, and by NSF grants NSF-0428075 and NSF-0325017.

[1]We do not treat the case of robot failure in this paper.

algorithm and demonstrates that the boundary coverage is complete. To illustrate the methodology, Section V applies the algorithm to several simulated environments. Open questions and possible extensions are summarized in Section VI.

II. The Boundary Coverage Problem

We consider a bounded 2-dimensional environment which is populated by N objects, $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_N$, an example of which is illustrated in Fig. 2. We collectively refer to the union of the boundaries of all objects as the "boundary" of the environment. The boundary of the i^{th} object is termed the "i^{th} boundary," and is denoted by $\partial\mathcal{O}_i$. We assume that the boundary of each object is a piecewise smooth, convex closed curve. A *boundary segment*, $[p_{i,1}, p_{i,2}]$ is the portion of $\partial\mathcal{O}_i$ between two points $p_{i,1}, p_{i,2} \in \partial\mathcal{O}_i$. The boundary segment $[p_{i,1}, p_{i,2}]$ includes the points between $p_{i,1}$ and $p_{i,2}$ inclusive on the boundary as it is traversed with freespace on the left (i.e., clockwise). We assume that the location and boundary geometry of each object is known a priori.

We assume that the boundary will be inspected by a group of k identical holonomic point robots, each equipped with an accurate scheme for localization as well as an omnidirectional or steerable "inspection sensor." Each inspection sensor has a maximum range of r. That is, each sensor is assumed to be capable of measuring the phenomena of interest when the robot is up to a distance r from the boundary. For the purposes of off-line planning, a point on the boundary is considered inspected when a robot has passed within distance r orthogonal to the boundary. For simplicity, our method assumes a preferred sensing distance of r, though the robot may inspect a boundary segment from a lesser distance. This constraint is imposed to make the construction of G easier by limiting the geometry of paths we initially consider to well-defined offset curves. We note that for robots with a steerable inspection sensor, an additional step is needed to determine the preferred or necessary sensor orientation at each instant during the robot's traversal of the inspection paths that are constructed below. This additional planning step to determine the sensor sweeping motions associated with a visibility path is relatively trivial, and therefore not discussed in detail in this paper.

All robots involved in the inspection task start from a common "depot" location at the beginning of the inspection period. At least one robot must inspect each point of the boundary at least once during the group's inspection tour. The robots' inspection routes are determined offline; they navigate through a series of waypoints determined by our method's solution. A secondary goal is to realize efficiency in the inspection task by balancing the work across the robots. In this paper, we seek to ensure that the path length of each robot's inspection tour is roughly equal.

To develop these paths, we take the sensor's visibility constraints into account. A boundary segment's *visibility space* consists of the set of individual robot poses whose distance to the nearest point on the boundary segment is less than r. A continuous sequence of poses within the visibility space is a *visibility path* if and only if every point in the boundary segment will have been viewed when a robot has assumed every pose in the visibility path.

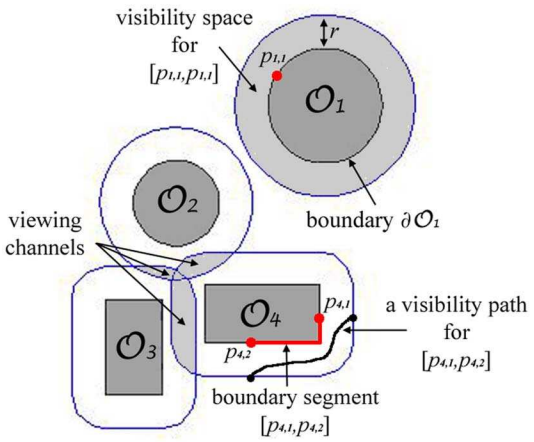

Fig. 2. Illustration of terminology introduced in Section II.

We refer to the region where the visibility spaces of disjoint boundary segments intersect as a *viewing channel*. From poses within a viewing channel, a robot can simultaneously inspect multiple boundary segments that are typically associated with different objects. Our routing strategy takes advantage of such channels, when they arise, as they may reduce the travel required for inspection by avoiding multiple passes through the same region.

III. A Graph Representation for the Boundary Coverage Problem

Our planning procedure starts by first constructing a graph representation of the inspection task. The edges of the graph come in two varieties: required inspection edges, E_R, and connectivity edges, E_{env}. Each required edge represents an equivalence class of visibility paths along which the robot must travel in order to inspect a boundary segment. A number of connecting edges are needed to connect the starting depot location with the inspection area and to provide paths between disconnected inspection regions. The bulk of the graph vertices represent a point of physical intersection of adjacent edges' path sets, where a robot may transition locally from a path represented in one edge to a path represented in another. The starting depot is an additional vertex. Additional vertices are added to provide efficient points of transit. The final result of the construction is an undirected, connected graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$, where V is the set of the graph's vertices, E is the set of its edges, and c is a cost function that assigns weights to the edges, in which $E_R \subseteq E$ is the set of edges whose traversal is required for inspection to be complete. The remaining edges $E_{env} = E \backslash E_R$ represent a collection of paths that connect required edges to one another and the depot location.

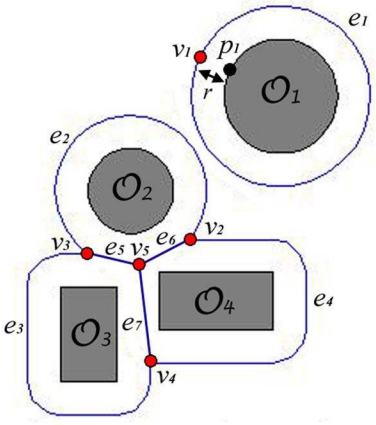

Fig. 3. The example illustrates the initial construction of G, in which edges disjoint from (e_1), adjacent to (e_2, e_3, and e_4), and inside (e_5, e_6, and e_7) viewing channels are constructed.

A. Determining E_R: Edges Required for Inspection

Each edge that must be traversed for inspection represents an equivalence class of paths which results in the same progress in the task, no matter which path in the set is taken. This representation allows flexibility in the navigation, as the individual can avoid one another by taking different paths while "traversing" the same graph edge. The end-points of each edge are graph vertices, which physically correspond to terminal points for the equivalence class of visibility paths.

The edges in E_R are constructed as follows:

1) Edges disjoint from a viewing channel: For each object boundary $\partial\mathcal{O}_i, i \in \{1, \ldots, N\}$ in the environment, we consider the visibility space of the entire boundary.

Lemma 3.1: If the visibility space of $\partial\mathcal{O}_i$ contains no viewing channels, traversal of the preferred visibility path, comprised of the perimeter of the visibility space, will result in the inspection of every point $p \in \partial\mathcal{O}_i$.

Proof: The preferred visibility path for $\partial\mathcal{O}_i$ is comprised of the perimeter of the object boundary's visibility space. If a robot traverses this path, defined by the offset curve a distance r from $\partial\mathcal{O}_i$, it will pass within distance r of every $p \in \partial\mathcal{O}_i$, resulting in the inspection of every point $p \in \partial\mathcal{O}_i$. ■

A vertex is placed on the preferred visibility path, a distance r from p_i, an arbitrarily chosen point on $\partial\mathcal{O}_i$. A single self-looping graph edge of weight equal to the length of the preferred visibility path, originating from and returning to v, is added to G. In Fig. 3, edge e_1 and vertex v_1 illustrate this type of edge addition.

2) Edges adjacent to a viewing channel: If the visibility space associated with $\partial\mathcal{O}_i$ incorporates one or more viewing channels, we divide $\partial\mathcal{O}_i$ into boundary segments, one associated with each viewing channel and one with each "non-channel" interval of $\partial\mathcal{O}_i$ that is not viewable from a point on the edge of $\partial\mathcal{O}_i$'s visibility space within the channel. By this construction, divisions of the boundary into segments occur at those points on the boundary that are a distance r from the intersection of the edge of $\mathcal{O}_i$'s visibility space with that of disjoint object boundary and a distance greater than r from all other object boundaries. The vertices of the edges associated with the viewing channel are determined as follows. The viewing channel boundaries are defined by the offset curves a distance r from the object boundaries. The point where the offset curves intersect is chosen as the terminal point of the viewing channel visibility paths. At these points, the robot is an equal distance r from both boundary segments contributing to the viewing channel. These points also define a terminal point of the adjacent non-channel edge. Examples of such vertices are v_2, v_3, and v_4 in Fig. 3.

Lemma 3.2: For each non-channel boundary segment, traversal of the preferred visibility path, comprised of the interval of the perimeter of the visibility space within distance r of the boundary segment, will result in the inspection of every point in that boundary segment.

Proof: By construction, the visibility path will pass within distance r of every point in the boundary segment, resulting in the inspection of every point in the boundary segment. ■

Each non-channel boundary segment in $\partial\mathcal{O}_i$ is assigned an edge connecting the vertices associated with its terminal points (see edges e_2, e_3, and e_4 in Fig. 3) of weight equal to the length of the edge of the visibility space between the those points, the preferred path for that edge.

3) Edges inside a viewing channel:

Lemma 3.3: All remaining points in the environment boundary $\partial\mathcal{O}_1 \cup ... \cup \partial\mathcal{O}_N$ lie in boundary segments which are visible from within a viewing channel.

Proof: By construction, the extreme points of each of these boundary segments, each of which is adjacent to the extreme point of a non-channel boundary segment, are a distance r from the intersection of the edge of the visibility space associated with the $\partial\mathcal{O}_i$ (of which that point is an element) with the visibility space of another $\partial\mathcal{O}_j, j \neq i$ and a distance $> r$ from all other object boundaries. Also by construction, every point not considered under Lemma 3.1 and Lemma 3.2 lies a distance less than$2r$ from a point on another object boundary, making it by definition viewable from within a viewing channel. ■

The robots are routed through viewing channels with a preference for paths whose constituent poses are equidistant from the boundaries currently being inspected. The preferred paths through the viewing channels are simply the local components of the *Generalized Voronoi Graph (GVG)* [4] of the nearby boundaries. The graph vertices at the end of the edges associated with viewing channels, by definition, lie on the GVG. Edges within viewing channels are established by adding to G all edges and vertices of the GVG contained within the union of all viewing channels in the environment. We note that GVG vertices within the union of all viewing channels occur where channels overlap. Edges only partially contained in a viewing channel are truncated where they meet a terminal vertex, which connects the viewing channel edge

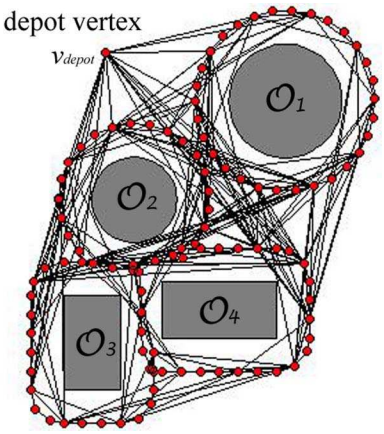

Fig. 4. A possible final graph representation of the example environment from Figs. 2 and 3.

to adjacent "non-channel" edges. For every point on the GVG within a viewing channel, points on at least two boundary segments can be sensed from a single robot pose in the omni-directional sensor case (in the steerable sensor case, a sensor rotation will be necessary, but can be carried out without changing the robot's location).

Lemma 3.4: Traversal of all edges of the GVG which lie within the viewing channels results in the inspection of all viewing channel boundary segments.

Proof: Every point on every edge of the GVG within a viewing channel is equidistant and $\leq r$ from the two nearest boundaries. Because no point on any viewing channel boundary segment is more than a distance $2r$ away from another object boundary, a point on a local GVG edge and contained within a viewing channel will fall within distance r of every point on the viewing channel boundary segment. It follows that traversal of that local GVG edge will result in that boundary segment's inspection. Thus, if all edges of the GVG which lie within viewing channels are traversed, every viewing channel boundary segment will be inspected. ■

Each edge e added to G from the GVG is assigned a weight equal to the length of the GVG path between the points represented by e's end vertices, the preferred path within a viewing channel. Edges $e_5, e_6,$ and e_7 in Fig. 3 illustrate this type of edge.

B. Determining E_{env}: Edges Providing Connectivity

Lone obstacles not associated with any viewing channels and clusters of obstacles connected by viewing channels will each comprise a connected componenet of G. After generating the edges in G that are necessary for boundary inspection, the possibly disjoint components of this construction must be connected.

1) Addition of vertices: To generate these connecting edges, the existing required edges are subdivided into a sequence of edges according to a user-selected *subdivision step size parameter*, while not changing the underlying graph geometry. These added *access vertices* offer opportunities to divide the inspection of the associated boundary segment among multiple robots, and some will also allow for more efficient transit to other parts of the graph. A vertex, v_{depot}, is added at the depot location at this time.

2) Addition of connecting edges: We then induce a *complete graph* on the vertices of G, adding edges connecting each vertex in G to every other vertex in G if such an edge is not already in place. We refer to these new edges as *induced edges*. Each represents a direct path through the environment from one vertex to another, and is assigned a weight equal to the distance between those vertices. If an induced edge's end vertices define a line segment that intersects a boundary, that edge is excluded from G, as it represents a path that does not lie entirely in freespace. All other induced edges are added to G and comprise the set of edges E_{env}.

If object boundaries $\partial\mathcal{O}_i$ and $\partial\mathcal{O}_j$, $i \neq j$, are within line of sight of one another, the subdivision step size must be small enough that $\partial\mathcal{O}_i$ has at least one vertex associated with its inspection lying in line of sight of at least one vertex associated with the inspection of $\partial\mathcal{O}_j$, meaning a direct path between them will lie in freespace. This will guarantee that E_{env} will include at least one edge joining each disjoint component of G to every other component within line of sight, thus guaranteeing the final graph will be connected.

3) Reducing graph size: the weeding parameter: As the number of edges in G becomes cumbersomely large, the constructive heuristic will require a long time to run to completion. To reduce the graph size while still achieving connectivity, the user may introduce a weeding procedure prior to the edge induction step. This step, which incorporates a user-selected *weeding parameter*, results in the inclusion of only a subset of vertices $V_{induce} \subseteq V$ in the edge induction process. The first vertex added to the initially empty set of vertices, V_{induce} is the depot vertex, v_{depot}. For every other vertex $v \in V$, if the shortest paths from v to every vertex in V_{induce} is greater than the distance specified by the weeding parameter, v is added to V_{induce}. Because the shortest path from a vertex within a disconnected graph component to any vertex outside the component is infinite, the construction of V_{induce} will result in the inclusion of at least one vertex from each disconnected component of G. The weeding parameter is subject to the same upper bound as the subdivision step size and must be sufficiently small to guarantee the graph will be connected by edges associated with paths lying in freespace.

A sample graph representation of our example environment from Figs. 2 and 3 is illustrated in Fig. 4.

IV. A Graph Algorithm for Solving the Boundary Coverage Problem

Based on the graph representation outlined in the last section, the k-robot boundary coverage problem reduces to a graph problem which we term the *k-Rural Postman Problem (kRPP)*. This problem is concerned with finding a set of $k \geq 1$ tours $T = \{T_1, ..., T_k\}$ within an undirected, connected weighted graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$ such

that each edge in a required subset of edges $E_R \subseteq E$ is traversed in at least one tour. Our nomenclature is adapted from a set of *Chinese Postman Problems* (CPPs) in the graph literature. Many variations of the CPP have been explored, of which the two most closely related to the graph problem at hand are:

1) the *Rural Postman Problem (RPP)*, an $NP-$hard problem in which the task is to find a minimum weight tour that traverses every edge in a required subset of edges in a connected, undirected graph G [5], and
2) the *Min-Max k-Chinese Postman Problem (MMkCPP)*, an $NP-$hard problem in which the task is to find a set of k tours of a connected, undirected graph G such that the weight of the longest tour is minimized, each tour starts and finishes at the same "depot" vertex, and every edge in G is traversed in at least one tour [6], [7].

We note that if $E_R = E$ the kRPP reduces to the kCPP; with the additional aim of minimizing the longest of the k tours, the MMkCPP represents such a case. In reference [6] this minimization is addressed both within the constructive heuristics they present and with post-construction tour improvement algorithms. While our formulation of the kRPP does not specifically require minimization of the longest tour, our heuristic does attempt to balance the inspection load spatially across the k tours by grouping neighboring edges into the same tours and seeks to avoid unnecessary redundant coverage. Like the RPP and MMkCPP, the problem we address is $NP-$hard. We present a constructive heuristic which finds a solution to the kRPP.

A. Constructive Heuristic for the kRPP

1) Definitions: Given an undirected, connected graph $G = (V, E, c : E \rightarrow \mathbb{R}^+)$ we denote the set of edges on the shortest path between vertices u and v, $\{u, v\} \in V$, $SP(u, v)$. The length of this path is denoted $C(SP(u, v))$.

The distance between a vertex v and an edge $e = \{x, y\}$ is defined as

$$d(v, e) = max\{C(SP(v, x)), C(SP(v, y)))\}. \quad (1)$$

We define the distance between two edges $e = \{x, y\}$ and $g = \{u, v\}$ as

$$d(e, g) = \quad max\{C(SP(u, x)), C(SP(u, y)), \\ C(SP(v, x)), C(SP(v, y))\}. \quad (2)$$

2) Cluster required edges: We begin by grouping the required edges E_R into k "clusters", $F_1 \cup ... \cup F_k = E_R$, using a farthest-point clustering method similar to that used in the "Cluster Algorithm" heuristic for the MMkCPP presented by Ahr and Reinelt [6], which is based on an algorithm which aims to minimize the maximum intra-cluster distance [8].

First, k *representative edges* $f_1, ..., f_k$ are found, where f_i is the first edge assigned to cluster F_i. The first representative edge f_1 is the edge $e \in E_R$ farthest from v_{depot} (as calculated with (1)) and is the first edge assigned to cluster F_1. The required edge with the maximum minimum distance (as calculated with (2)) to the existing representative edges $f_1, ..., f_{i-1}$ is called f_i and is assigned to F_i. The $|E_R| - k$ remaining edges $e \in E_R$ are assigned to the cluster F_i that minimizes the distance between e and f_i.

3) Include edges for connectivity: The task remains to ensure each cluster of edges is connected and, thus, traversable. Given the subgraph $G_{R_i} = G[F_i + v_{depot}]$, we construct a graph G'_{R_i} with a vertex representing each connected component of G_{R_i}. An edge is added between every vertex u and v. The weight $c(e)$ of edge $e = \{u, v\}$ is equal to the length of the shortest path on G from any vertex in component u to component v. A minimum spanning tree is computed on G'_{R_i}; the edges in the shortest path in G associated with each edge in the spanning tree are added to F_i.

4) Compute a tour of each cluster: Next, a single-postman tour T_i is computed on the subgraph $G[F_i], i = \{1, ..., k\}$ using Edmonds' and Johnson's CPP algorithm [9]. Each tour $T_i \in \{T_1, ..., T_k\}$ originates and terminates at v_{depot}, and has length $C(T_i) = \sum_{e \in T_i} c(e)$.

5) Improve the tours: Finally, for each cluster F_i, we consider the subset of required edges, F_{i_R}. While traversing T_i, after each edge $e \in F_{i_R}$ has been visited at least once, the remaining edges in T_i are replaced with the shortest path on G back to v_{depot}.

B. Path Planning Using the Graph Solution

The k robots' inspection paths are planned based on the graph solution to the kRPP. Because edges in G specify preferred paths, the tours found by the graph algorithm each define a series of waypoints. Each robot is assigned a tour and visits that tour's waypoints in sequence. When robots must pass within close proximity of one another in simulation, priority is established through a simple leader election process, in which priority is given to the robot with the largest distance left to travel. Other robots wait until the leader proceeds out of their proximity. A more sophisticated approach to this final path planning step will be addressed in future work.

C. Completeness of Boundary Coverage

Lemma 4.1: The set of k tours generated by the constructive heuristic guarantees each edge in the required subset of edges E_R is traversed in at least one tour.

Proof: By construction, every edge $e \in E_R$ is assigned to a cluster. Edges are added to each cluster to ensure each is a connected subgraph of G. A of tour each cluster is then calculated with the Edmonds and Johnson CPP algorithm, an algorithm which guarantees every edge in the cluster will be traversed. As every edge in E_R is contained in a cluster, and every edge in every cluster will be traversed by a tour, it follows that every edge in E_R will be traversed in at least one tour. ■

Proposition 4.2: The paths planned based on the kRPP graph solution provide complete coverage of the test environment's boundary.

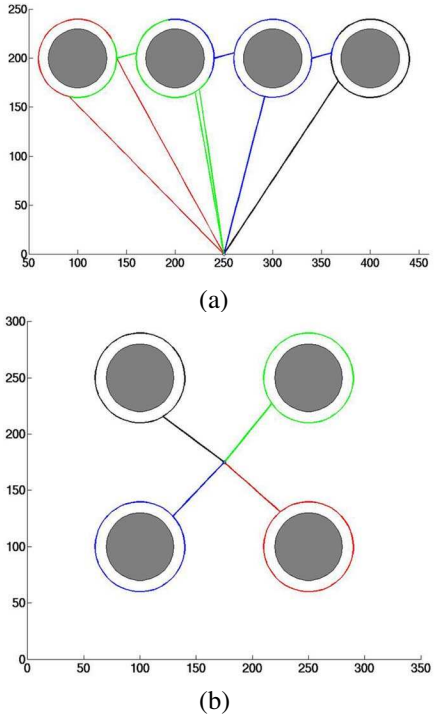

Fig. 5. The environments in (a) and (b) contain four identical circular boundaries, but placement has a large effect on the "fairness" of the planned routes. The routes planned for $k = 4$ robots are illustrated with waypoints marked by a route-specific colors of black, green, red, and blue.

Proof: In Section III-A, we consider the visibility space of each object boundary $\partial\mathcal{O}_i$ in turn. By Lemmas 3.1, 3.2, and 3.3, when every $\partial\mathcal{O}_i$ has been considered, every point in the boundary has been considered as a constituent point in either a disjoint closed boundary segment, a viewing channel segment, or a "non-channel" segment. Graph edges representing preferred visibility paths for each of these segments have been constructed. If each of these preferred visibility paths is traversed, every point on the boundary will have been inspected; boundary coverage will be complete. By Lemma 4.1, the kRPP heuristic gives a graph solution that guarantees the traversal of every edge in E_R. The traversal of a graph edge corresponds to the traversal of the associated preferred visibility path, guaranteeing complete boundary coverage. ■

V. Results and Discussion

In this section we provide three examples that illustrate some of the key characteristics of our algorithm and some of the main issues related to its practical use. We note that the distance units used in simulation are arbitrary and are used to show relative distance.

A. Division of Labor by Clustering Edges

This first examples illustrates the division of labor imposed by the kRPP approach to routing the robots. In this example, four robots are tasked to inspect an environment containing four identical circular boundaries, and robot's sensing range r is small enough that no viewing channels arise. Applying our algorithm to the configuration of the disks shown in Fig. 5a, results in a counterintuitive division of the task where one robot is not assigned to individually inspect each disk boundary. This somewhat unusual division arises from the nature of the farthest-point edge clustering method used to divide the edges among the k tours. While there are four distinct boundaries, the clustering algorithm may group the edges in seeming unnatural ways. When the boundaries are arranged in an equidistant pattern from the depot and are sufficiently separated so that their inspection edges cluster in a natural way, then an "intuitive" division of labor arises, as illustrated in 5b.

While the routes are of equal length in this case, the scenario in Fig. 5a, in comparison, yielded a longest route 38% longer than the shortest route and 24% longer than the average length of the four routes. Though the paths are not intuitive in that scenario, the load is still roughly balanced among the robots. Because the clustering algorithm seeks to minimize the intra-cluster distance of each cluster, it does not "fairly" distribute the weight of the edges in each group; how evenly the inspection load is balanced across the robots will be dependent on the weight and concentration of required edges in proximity to the k representative edges. For an inspection with a relatively homogeneous distribution of required edges, the routing will, however, certainly be "fair" enough to offer a tour length advantage with $k > 1$ robots.

Further tour weight optimization with alternative clustering methods and post-construction tour improvement heuristics is possible in many instances and will be considered in future work.

B. Viewing Channels and More Complex Environments

We have chosen two illustrative examples of routing through more complex environments. The first "scattered" environment contains several objects of various sizes and shapes, while the second "packed" environment contains a closely spaced set of identical objects. For each environment, we consider two values of r, one small enough that no viewing channels arise, the other large enough that every boundary has an associated viewing channel. The scattered environment is shown in Fig. 6, while the packed environment is illustrated in Fig. 7.

The major difference between the two environment examples is the portion of the edges in E_R that lie inside viewing channels. If a large portion of the edges E_R are inside viewing channels, their weight will not change as significantly as external edges' weights will as we increase r. In the scattered environment, the increase in length of the preferred paths as r increases outweighs the benefit in path reduction of traversing a viewing channel. The tours in the large-r scattered environment example tend to be longer than in the small-r case because the total weight of edges in E_R was greater than in the small-r case, inflation resulting from the relatively small portion of E_R associated with viewing channels. We note that a large part of this disadvantage due to inflation is artificially imposed by the constraint of the preferred path's distance r

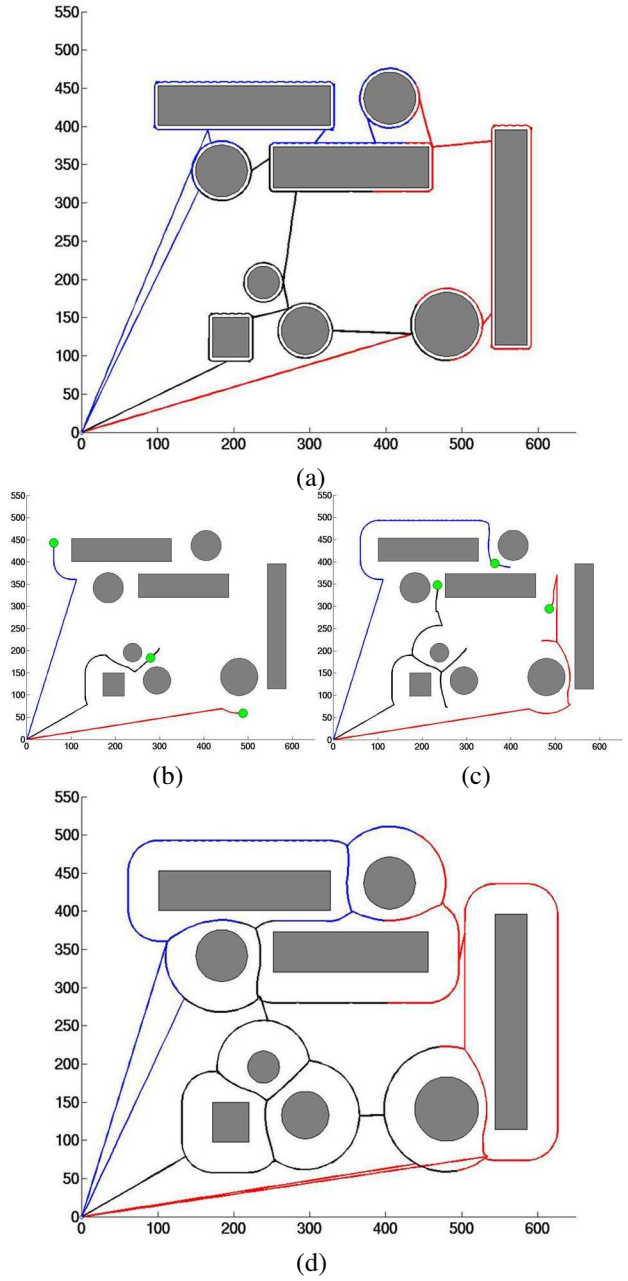

Fig. 6. In an environment with several objects of various sizes and shapes, we illustrate routes planned for $k = 3$ robots with (a) $r = 5$, small enough that no viewing channels arise, and ((b), (c), and (d)) $r = 40$, large enough every boundary is associated with a viewing channel. Routes are marked by a route-specific color of blue, black, or red. Two "snapshots" of the $r = 40$ inspection in progress are shown at time steps (b) 50 and (c) 100. A total of 605 time steps are required for the inspection to be completed and for all robots to return to the depot. The green dots in (b) and (c) represent the robots' positions at that given time step.

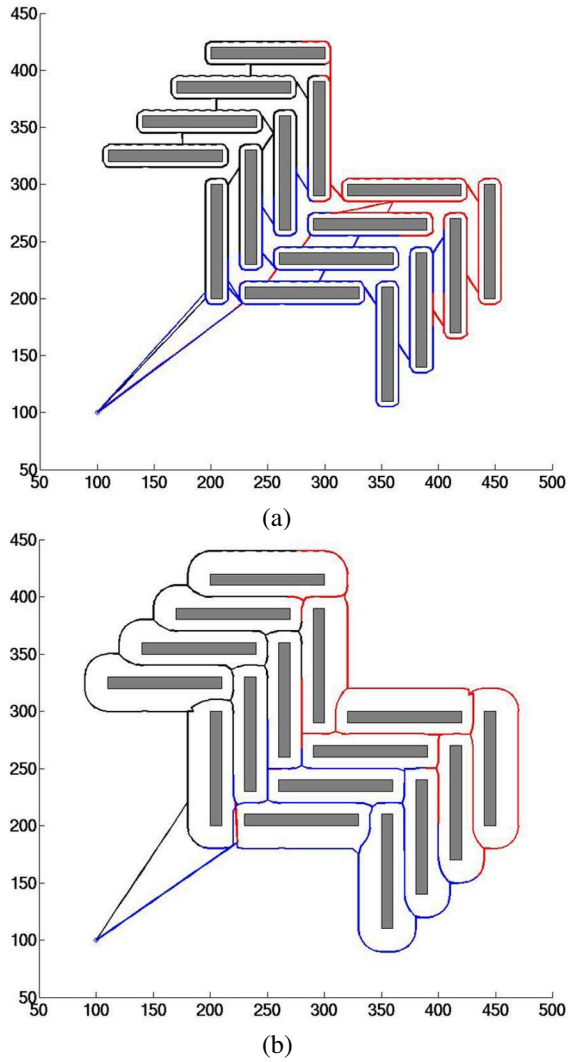

Fig. 7. In an environment with identical, closely packed objects, we illustrate routes planned for $k = 3$ robots with (a) $r = 5$, small enough that no viewing channels arise, and (b) $r = 20$, large enough most edges in E_R are part of a viewing channel. Routes are marked by a route-specific color of black, blue, or red.

from the boundary. The constraint will be relaxed in future work as we explore options for pre- and post-routing local path planning and length optimization.

The packed environment is reminiscent of the turbine-blade environment (Fig. 1) that motivated this inspection task. In this example we observe the path-length advantage of exploiting viewing channels in our routing task. For large enough r in a crowded enough environment, a large portion of the edges in E_R will be within viewing channels, allowing the routing algorithm to avoid multiple passes through the same region. The weight of edges within viewing channels will not change as significantly as external edges' weights will as r increases. The packed environment in Fig. 7 illustrates the advantage of exploiting viewing channels in a packed environment, as the large-r tour lengths are significantly shorter than the small-r tours, as shown in Fig. 9.

In both examples, despite the difference in graph structure between the two r-value cases, the resulting routes are remarkably similar in their division of the task. This is evident in Figs. 6 and 7; each route in the small-r scenario has a corresponding route in the large-r scenario that inspects approximately the same set of boundary segments.

The effects of team size k on tour lengths are shown in Figs. 8 and 9. Despite the pronounced difference in graph structure introduced by the different r values, the

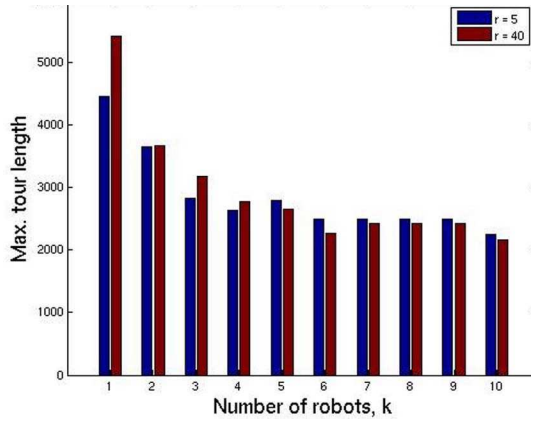

Fig. 8. Comparison of the longest tour lengths vs. number of robots k for $r = 5$ and $r = 40$ in the "scattered" environment in Fig. 6.

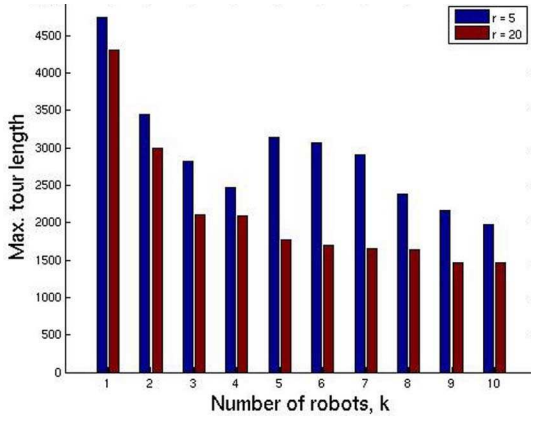

Fig. 9. Comparison of the longest tour lengths vs. number of robots k for $r = 5$ and $r = 20$ in the "packed" environment in Fig. 7. We note that for $r = 20$, the tour lengths are significantly shorter than for the $r = 5$ case, illustrating the advantage of exploiting viewing channels in a packed environment.

performance is qualitatively similar for both example environments. With the addition of a few robots, the longest tour length drops dramatically, but the tour length benefit decreases exponentially with k. As k increases, the average tour length levels off. This is because the constructive heuristic requires that every tour contains at least one required edge (due to the clustering method), and as k increases, the heuristic begins developing nearly identical tours for multiple robots.

For a cost function based on an time to completion, a choice in which the path length of the longest tour will dominate, dividing the task among two or more robots offers a marked benefit, but if the cost function is primarily based on total robot energy output, e.g., distance traveled, using a single robot will likely offer a better strategy.

VI. Conclusions and Future Work

This paper introduced the multi-robot boundary coverage problem, formulated a graph representation of the problem, and posed the associated planning problem as the k-Rural Postman Problem. Because the exact solution to this problem is NP-hard, we developed a constructive heuristic which finds a solution to the kRPP, and then used that solution to plan the robots' inspection routes. The solution provides complete coverage of the environment boundary and seeks to divide the inspection load amongst the k robots.

The diminishing returns in performance relative to tour length are largely due to our restriction that the cooperating robots must disperse from and return to the depot along paths that have already been traveled. A logical next step in applying the graph representation presented here will be to pose a task better suited to the features of a graph-based solution, such as a surveillance task requiring periodic re-inspection, minimizing returns to the depot and maximizing the benefit of revisiting edges.

We hope to further develop the application of graph methods to boundary coverage problems, exploring graph representations in terms of cost functions based on total distance, consumption of power, memory limits, and reliability of data in addition to the longest tour performance measures used in this paper. Future work should also include waypoint following, handling of robot-robot interference, local path optimization, and variable team size. To implement a similar method on physical robots, we will need to begin relaxing our constraints, allowing for imperfect sensing, localization, imperfect prior knowledge of the environment, and finite robot size.

Acknowledgments

The authors would like thank Elon Rimon and Eric Klavins for valuable discussions regarding graph-based approaches to multi-robot coverage and Howie Choset for the use of his planar GVG construction software, which was the basis for the graph construction step in our simulations.

References

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

[2] Y. Zhang, E. K. Antonsson, and A. Martinoli, "Evolving neural controllers for collective robotic inspection," *Proc. of the 9th Online World Conf. on Soft Computing in Industrial Applications*, 2004.

[3] N. Correll and A. Martinoli, "Collective inspection of regular structures using a swarm of miniature robots," *Proc. of the Ninth Int. Symp. on Experimental Robotics ISER-04*, 2004.

[4] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[5] C. S. Orloff, "A fundamental problem in vehicle routing," *Networks*, vol. 4, pp. 35–64, 1974.

[6] D. Ahr and G. Reinelt, "New heuristics and lower bounds for the min-max k-chinese postman problem," in *Algorithms-Esa 2002, Proceedings*, ser. Lecture Notes in Computer Science, 2002, vol. 2461, pp. 64–74.

[7] G. N. Frederickson, M. S. Hecht, and C. E. Kim, "Approximation algorithms for some routing problems," *SIAM Journal on Computing*, vol. 7, no. 2, pp. 178–193, 1978.

[8] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, no. 2-3, pp. 293–306, 1985.

[9] J. Edmonds and E. L. Johnson, "Matching, euler tours, and the chinese postman," *Mathematical Programming*, vol. 5, pp. 88–124, 1973.