49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

Multi-resolution Path Planning: Theoretical Analysis, Efficient Implementation, and Extensions to Dynamic Environments

Raghvendra V. Cowlagi and Panagiotis Tsiotras

Abstract— **A multi-resolution path planning algorithm based on the wavelet transform of the environment has been reported previously in the literature. In this paper, we provide a proof of completeness of this algorithm. In addition, we present an implementation of this algorithm that reuses information obtained in previous iterations to perform subsequent iterations more efficiently. Finally, we extend this path planning algorithm to dynamic environments by presenting a simple scheme for updating the wavelet transform coefficients to reflect changes in the environment.**

I. INTRODUCTION

Motion planning for autonomous mobile vehicles is the problem of finding control inputs that enable the vehicle's motion to satisfy some pre-assigned task [1], [2]. Due to its inherent complexity, the motion planning problem is usually solved over two levels of hierarchy. The higher level, called the geometric path planning level, deals with obstacle avoidance by finding an obstacle-free *path* from the initial point to the destination. The lower layer deals with the kinematical and dynamical constraints of the vehicle by generating a feasible reference *trajectory* based on the path found by the geometric planner.

Geometric path planning based on cell decompositions [1, Ch. 5] is a widely used technique that involves partitioning the environment into convex, non-overlapping regions called *cells*. A graph is then associated with the cell decomposition, where each obstacle-free cell is represented by a node in the graph, and the geometric adjacency relationships of the cells are represented by edges. A path in this graph then corresponds to a sequence of obstacle-free cells from the initial cell to the goal cell.

In this paper, we discuss a geometric path planning scheme based on *multi-resolution* cell decompositions. The quadtree method [3]–[5], which employs dyadic recursive decompositions, is one of the most extensively used multi-resolution cell decomposition techniques. Other path planning schemes using multi-resolution cell decompositions have been proposed in [6]; [7] (triangular cells); [8] (receding horizon path planning using multi-resolution estimates of object locations); [9] (multi-resolution potential field); and [10] (hierarchy of imaginary spheres encapsulating the robot, for collision avoidance).

References [11], [12] report a new hierarchical path planning algorithm using multi-resolution cell decompositions obtained via the wavelet transform of the obstacle space. We describe this algorithm in greater detail in subsequent sections of this paper. Other applications of the wavelet transform to multi-resolution path planning schemes appear in [13]–[15].

This work extends the wavelet-based path planning scheme of [11] with three new results: first, we provide a proof of completeness of (a slightly modified version) the path planning algorithm in [11]; second, we provide a method to efficiently recompute the multi-resolution cell decomposition and its associated topological graph by suitably modifying the graph computed in the previous iteration (as opposed to computing it from scratch at each iteration, as is done [11], [12]); and third, we provide a method to efficiently update the wavelet coefficients to match changes in the environment.

In the next section, we provide a cursory and informal introduction to the wavelet transform; for further details the reader may consult, for instance, [16].

A. Multi-resolution Cell Decompositions

Multi-resolution analysis of a scalar function involves the construction of a hierarchy of approximations of the function by projecting it onto a sequence of nested linear spaces. In the context of the wavelet transform, these linear spaces are the spans of translated and scaled versions of two functions called, respectively, the *scaling function* and *mother wavelet*. The *discrete wavelet transform* of a function $F \in \mathcal{L}^2(\mathbb{R})$ (or, in the 2-D case, a function $F \in \mathcal{L}^2(\mathbb{R}^2)$) then refers to two collections of scalars, called, respectively, the *approximation* coefficients and the *detail* coefficients, defined, respectively, as the inner products of F with translated and scaled versions of the scaling function and the mother wavelet.

To apply the discrete wavelet transform for the multi-resolution analysis of the obstacle space, we define an *image* as the pair (R, F), where $R \subset \mathbb{R}^2$ is a compact, square region and $F : R \to \mathbb{R}_+$ is an intensity map[1]. We assume that $R = \left[0, 2^D\right] \times \left[0, 2^D\right]$, with $D \in \mathbb{Z}$. We also assume that the image intensity map F is known at a finite maximum resolution $j_{\max} > -D$, i.e., the function F is piecewise constant over each of the square cells $\left[2^{-j_{\max}}k, 2^{-j_{\max}}(k+1)\right] \times \left[2^{-j_{\max}}\ell, 2^{-j_{\max}}(\ell+1)\right]$, $k, \ell = 0, 1, \ldots, 2^{D-j_{\max}} - 1$.

A *cell decomposition* $\mathcal{C}$ of an image is a finite collection of disjoint, convex subsets C_n of R called *cells*, such that $\bigcup_{n=1}^{N_\mathcal{C}} C_n = R$, and F is constant over the cell C_n, for each $n = 1, 2, \ldots, N_\mathcal{C}$, where $N_\mathcal{C} \in \mathbb{N}$. In this paper, we consider square cells, and we denote by $C(j, k, \ell)$ a cell of size 2^{-j} units, with its center at the

R. V. Cowlagi is a Ph.D. candidate at the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. e-mail: rcowlagi@gatech.edu

P. Tsiotras is with the Faculty of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. e-mail: tsiotras@gatech.edu

[1]In the context of path planning, an image may represent, for instance, a terrain elevation map, or a risk measure of the environment [11].

point $(2^{-j}k+2^{-j-1}, 2^{-j}\ell+2^{-\ell-1})$. The collection of cells $\mathcal{C}_j \overset{\text{def}}{=} \{C(j,k,\ell) : k,\ell = 0,1,\ldots,2^{D-j}-1\}$ is a *uniform* cell decomposition of the image (R,F) at resolution level j.

We associate with the cell decomposition $\mathcal{C}_{j_{\max}}$ a graph $\bar{\mathcal{G}} \overset{\text{def}}{=} (\bar{V},\bar{E})$, such that each node in $\bar{V}$ corresponds to a unique cell in $\mathcal{C}_{j_{\max}}$. Two nodes in $\bar{V}$ are *adjacent* if the corresponding cells in $\mathcal{C}_{j_{\max}}$ are geometrically adjacent[2]. The edge set $\bar{E}$ consists of all pairs $(\bar{u},\bar{v})$, where $\bar{u},\bar{v} \in \bar{V}$ are adjacent nodes. We use $\mathsf{cell}(\bar{u};\mathcal{C}_{j_{\max}})$ to denote the cell in $\mathcal{C}_{j_{\max}}$ corresponding to the node $\bar{u} \in \bar{V}$; conversely, we denote by $\mathsf{node}(C;\bar{\mathcal{G}})$ the node in $\bar{V}$ corresponding to the cell $C \in \mathcal{C}_{j_{\max}}$. Finally, we use $(x(\bar{u}),y(\bar{u}))$ for the coordinates of the center of the cell corresponding to node $\bar{u}$. We introduce an edge cost function $\bar{g} : \bar{E} \to \mathbb{R}_+$, which assigns to each edge of $\bar{\mathcal{G}}$ a non-negative cost of transitioning this edge. For given initial and terminal nodes $\bar{u}_{\mathrm{S}}, \bar{u}_{\mathrm{G}} \in \bar{V}$, a *path* $\bar{\pi}(\bar{u}_{\mathrm{S}},\bar{u}_{\mathrm{G}}) \overset{\text{def}}{=} (\bar{u}_0^{\bar{\pi}}, \bar{u}_1^{\bar{\pi}}, \ldots, \bar{u}_{\bar{P}}^{\bar{\pi}})$ in $\bar{\mathcal{G}}$ is such that $\bar{u}_k^{\bar{\pi}} \in \bar{V}$, $(\bar{u}_{k-1}^{\bar{\pi}}, \bar{u}_k^{\bar{\pi}}) \in \bar{E}$, $k=1,\ldots,\bar{P}$, with $\bar{u}_0^{\bar{\pi}} = \bar{u}_{\mathrm{S}}$, $\bar{u}_{\bar{P}}^{\bar{\pi}} = \bar{u}_{\mathrm{G}}$, and $\bar{u}_p^{\bar{\pi}} \neq \bar{u}_r^{\bar{\pi}}$, for $p,r \in \{0,\ldots,\bar{P}\}$, with $p \neq r$. For brevity, and when there is no ambiguity, henceforth we will suppress the arguments in $\bar{\pi}$. The cost of a path $\bar{\pi}$ is defined as $\bar{\mathcal{J}}(\bar{\pi}) \overset{\text{def}}{=} \sum_{k=1}^{\bar{P}} \bar{g}((\bar{u}_{k-1}^{\bar{\pi}}, \bar{u}_k^{\bar{\pi}}))$. The *path planning problem* is to find a path $\bar{\pi}^*(\bar{u}_{\mathrm{S}},\bar{u}_{\mathrm{G}})$ such that $\bar{\mathcal{J}}(\bar{\pi}^*) \leqslant \bar{\mathcal{J}}(\bar{\pi})$ for every path $\bar{\pi}$ in $\bar{\mathcal{G}}$.

Algorithms such as Dijkstra's algorithm and the A* algorithm [2] can efficiently solve the path planning problem described above. However, the graph $\bar{\mathcal{G}}$ associated with large environments (i.e., with D large) consists of a very large number of nodes. For practical, real-time applications, the execution of standard search algorithms on the graph $\bar{\mathcal{G}}$ may be time-consuming; it may also be unnecessary, because the vehicle's path needs to be known accurately only in the immediate vicinity of the vehicle. In light of this observation, [11] proposes a path planning algorithm based on *multi-resolution* cell decompositions, constructed as follows.

Let a coarse resolution level $j_{\min} \in \mathbb{Z}$ be specified, and for $j \geqslant j_{\min}$, let $a_{j_{\min},k,\ell}$ and $d_{j,k,\ell}^i$ $(i=1,2,3)$ be the 2-D discrete wavelet transform coefficients of the map F. Let $\mathcal{A} \overset{\text{def}}{=} \{(j_m,k_m,\ell_m)\}$ be a set of triplets of integers such that $j_m \geqslant j_{\min}$, $m \in \mathbb{N}$. An *approximation* (R,F^{mr}) is the image obtained by the coefficients $a_{j_{\min},k,\ell}$ and $\hat{d}_{j,k,\ell}^i$, where

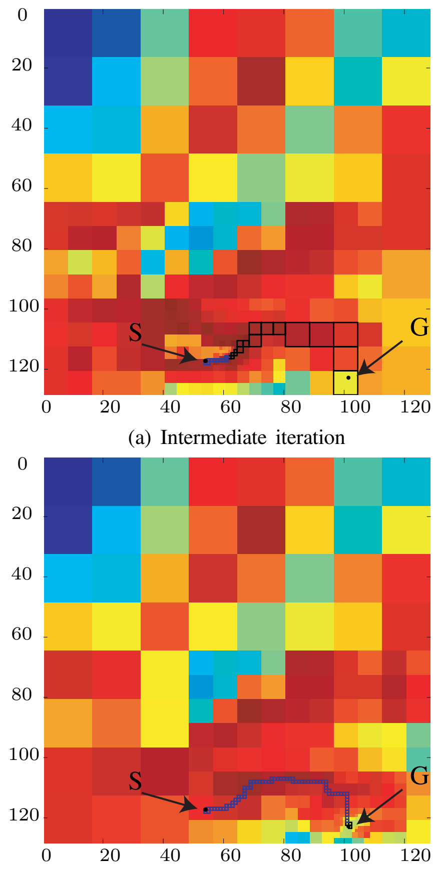
$$\hat{d}_{j,k,\ell}^i \overset{\text{def}}{=} \begin{cases} d_{j,k,\ell}^i & i=1,2,3;\ (j,k,\ell) \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases}$$

In what follows, we will refer to the approximation F^{mr} of F by its associated set of detail coefficients $\mathcal{A}$. Each approximation F^{mr} uniquely corresponds to a cell decomposition $\mathcal{C}^{\mathrm{mr}}$ that consists of cells of different sizes. We associate with $\mathcal{C}^{\mathrm{mr}}$ a graph $\mathcal{G} = (V,E)$ such that each node in V corresponds to a unique cell in $\mathcal{C}^{\mathrm{mr}}$. Each node $u \in V$ also corresponds to a set $W(u,V) \overset{\text{def}}{=} \{\bar{w}_0^u, \bar{w}_1^u, \ldots, \bar{w}_{S(u)}^u\} \subset \bar{V}$, such that $\{W(u,V)\}_{u\in V}$ is a partition of $\bar{V}$. The multi-resolution cell decomposition graph $\mathcal{G}$ approximates the graph $\bar{\mathcal{G}}$ by representing each set of nodes $W(u,V) \subset \bar{V}$ with a single node $u \in V$. For the Haar wavelet, it can be shown that for $u \in V$,

$$F^{\mathrm{mr}}(\mathsf{cell}(u;\mathcal{C}^{\mathrm{mr}})) = \frac{1}{S(u)+1} \sum_{m=0}^{S(u)} F(\mathsf{cell}(\bar{w}_m^u;\mathcal{C}_{j_{\max}})). \tag{1}$$

Finally, two nodes $u,v \in V$ are said to be adjacent in $\mathcal{G}$, i.e., $(u,v) \in E$, if and only if there exist $\bar{u} \in W(u,V)$ and $\bar{v} \in W(v,V)$ such that $(\bar{u},\bar{v}) \in \bar{E}$.

[2]Here we assume 4-connectivity, i.e., cells with two vertices in common are said to be adjacent. This assumption implies that $|\bar{E}| < 4|\bar{V}|$.



(a) Intermediate iteration

(b) Terminal iteration

Fig. 1. Illustration of the path planning algorithm.

II. Multi-Resolution Path Planning

Informally, the multi-resolution path algorithm of [11] operates as follows: at each iteration, a multi-resolution cell decomposition is constructed. This decomposition retains high resolution in the immediate vicinity of the vehicle's current location, and approximates the environment in regions farther away. A standard search algorithm may then be executed quickly on the graph $\mathcal{G}$ because $|V| \ll |\bar{V}|$.

Figure 1(a) illustrates an intermediate iteration in the path planning algorithm: the black colored cells indicate the optimal path found at that iteration, while the blue colored cells indicate the path recorded until that iteration. Figure 1(b) illustrates the final step, with a cell decomposition consisting of high resolution cells in the vicinity of the goal. The blue cells indicate the path found by the algorithm from the initial node to the goal node.

Next, we present a modification of the multi-resolution path planning algorithm of [11], and we prove that this modified algorithm is complete. To this end, we assume that the environment consists of free space and insurmountable obstacles, i.e., $F(\mathsf{cell}(\bar{u};\mathcal{C}_{j_{\max}})) = 0$ if $\bar{u}$ represents free space, and $F(\mathsf{cell}(\bar{u};\mathcal{C}_{j_{\max}})) = M$ if $\bar{u}$ represents an obstacle, where $M \gg |\bar{V}|$. We define the transition cost of an edge $(\bar{u},\bar{v}) \in \bar{E}$ by

$$\bar{g}((\bar{u},\bar{v})) \;=\; F(\bar{v})+1, \quad (\bar{u},\bar{v}) \in \bar{E}. \tag{2}$$

To the multi-resolution approximation of the environment constructed at iteration n of the algorithm, we let $\mathcal{A}(n)$ denote the associated set of detail coefficients, $\mathcal{C}^{\mathrm{mr}}(n)$ denote the associated multi-resolution cell decomposition, and $\mathcal{G}(n) = (V(n), E(n))$ denote the associated topological graph. We define the goal node $u_{\mathrm{G},n} \in V(n)$ as the (unique) node that satisfies $\bar{u}_{\mathrm{G}} \in W(u_{\mathrm{G},n}, V(n))$.

For each node $\bar{u} \in \bar{V}$, the proposed algorithm maintains an estimate $J(\bar{u})$ of the least cost of any path in $\bar{\mathcal{G}}$ from the node $\bar{u}$ to the goal node $\bar{u}_{\mathrm{G}}$, and a record $K(\bar{u})$ of the least cost of any path in $\bar{\mathcal{G}}$ from the initial node $\bar{u}_{\mathrm{S}}$ to the node $\bar{u}$. The algorithm also associates with each node $\bar{u} \in \bar{V}$ another node $B(\bar{u}) \in \bar{V}$ called the *backpointer* of $\bar{u}$. At each iteration, the algorithm performs a computation (specifically, in Line 17 or Line 19 of procedure MAIN below) whose result is a unique node in $\bar{V}$. We refer to this computation as a ***visit*** to this node, and we denote by $\bar{u}_n$ the node visited by the algorithm at iteration $n \in \mathbb{N}$, with $\bar{u}_0 \overset{\mathrm{def}}{=} \bar{u}_{\mathrm{S}}$. Let $u_n \overset{\mathrm{def}}{=} \mathsf{node}(\mathsf{cell}(\bar{u}_n;\mathcal{C}^{\mathrm{mr}}(n));\mathcal{G}(n))$, i.e., u_n is the node in $V(n)$ corresponding to the cell at the finest resolution represented by the node $\bar{u}_n \in \bar{V}$.

A path $\pi_n(u_n, u_{\mathrm{G},n}) = (u_0^{\pi_n}, u_1^{\pi_n}, \ldots, u_{P(n)}^{\pi_n})$ in $\mathcal{G}(n)$ is such that $u_p^{\pi_n} \neq \mathsf{node}(\mathsf{cell}(B(\bar{u}_n);\mathcal{C}^{\mathrm{mr}}(n));\mathcal{G}(n))$, and $u_p^{\pi_n} \neq w_m$ for each $p \in \{0,\ldots,P(n)\}$, and for each $m = 0,1,\ldots,n-1$, where $w_m \in V(n)$ is the unique node that satisfies $\bar{u}_m \in W(w_m, V(n))$. Note that this definition precludes cycles in the path in $\mathcal{G}(n)$ obtained by the concatenation of the path $(w_0, w_1, \ldots, w_{n-1})$ with π_n. The cost $\mathcal{J}_n(\pi_n)$ of the path $\pi_n(u_n, u_{\mathrm{G},n})$ is

$$\mathcal{J}_n(\pi_n) \overset{\mathrm{def}}{=} \sum_{m=1}^{P(n)} g_n((u_{m-1}^{\pi_n}, u_m^{\pi_n})). \tag{3}$$

The transition cost function $g_n : E(n) \to \mathbb{R}_+$ in (3) is

$$g_n(u,v) \overset{\mathrm{def}}{=} M\delta(F^{\mathrm{mr}}(C^v) - M) + |W(v, V(n))|, \tag{4}$$

where $(u,v) \in E(n)$, $C^v \overset{\mathrm{def}}{=} \mathsf{cell}(v;\mathcal{C}^{\mathrm{mr}}(n))$, and $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ otherwise. Note that, by (4), the cost of an obstacle-free path in $\mathcal{G}(n)$ is less than or equal to the number $|\bar{V}|$ of nodes in the graph $\bar{\mathcal{G}}$, and hence a path π_n in $\mathcal{G}(n)$ is obstacle-free if and only if $\mathcal{J}_n(\pi_n) < M$.

The algorithm associates with each node $\bar{u} \in \bar{V}$ a binary value VISITED($\bar{u}$), which records whether the node $\bar{u}$ has previously been visited by the algorithm, i.e., at any iteration of the algorithm's execution, and for any $\bar{u} \in \bar{V}$, VISITED($\bar{u}$) $= 0$ indicates that the algorithm has never visited $\bar{u}$ in any previous iteration, whereas VISITED($\bar{u}$) $= 1$ indicates that the algorithm has visited $\bar{u}$ at least once during previous iterations. The algorithm also maintains a cumulative cost $\bar{\mathcal{J}}(\bar{\pi})$ of the path $\bar{\pi}(\bar{u}_{\mathrm{S}}, \bar{u}_n)$ in $\bar{\mathcal{G}}$. Finally, to construct approximations that retain the detail coefficients in a "window" centered at the agent's location, we introduce a function $\varrho : \mathbb{Z} \to \mathbb{N}$ that associates with each resolution j the size of the "window" at that resolution. The multi-resolution path planning algorithm is then described as follows.

```
procedure MR-Approx($\bar{u}$)
1: $\mathcal{A} \leftarrow \Big\{ d^i_{j,k,\ell} : j_{\min} \leqslant j < j_{\max},\ i = 1,2,3,$
       $\lfloor 2^j x(\bar{u}) \rfloor - \varrho(j) \leqslant k \leqslant \lfloor 2^j x(\bar{u}) \rfloor + \varrho(j),$
       $\lfloor 2^j y(\bar{u}) \rfloor - \varrho(j) \leqslant \ell \leqslant \lfloor 2^j y(\bar{u}) \rfloor + \varrho(j) \Big\}.$
procedure Main()
1: $\bar{\pi} \leftarrow \bar{u}_{\mathrm{S}}$, $\bar{u}_0 \leftarrow \bar{u}_{\mathrm{S}}$, $n \leftarrow 0$, reachedGoal $\leftarrow 0$, $\bar{\mathcal{J}}(\bar{\pi}) \leftarrow 0$
2: For each $\bar{u} \in \bar{V}$, VISITED($\bar{u}$) $\leftarrow 0$
3: while !reachedGoal and $\bar{\mathcal{J}}(\bar{\pi}) < M$ and $J(\bar{u}_n) < M$ do
4:   $\mathcal{A}(n) \leftarrow$ MR-APPROX($\bar{u}_n$),
5:   $\mathcal{G}(n) \leftarrow$ MR-GRAPH($\mathcal{A}(n)$)
6:   if VISITED($\bar{u}_n$) $= 1$ then
7:     $\pi^*_n \leftarrow \arg\min\{\mathcal{J}_n(\pi) : \pi$ obstacle-free in $\mathcal{G}(n)\}$,
       subject to $\mathcal{J}_n(\pi^*_n) \geqslant J(\bar{u}_n) + 1$
8:     $\bar{\mathcal{J}}(\bar{\pi}) \leftarrow K(\bar{u}_n)$
9:   else
10:    $\pi^*_n \leftarrow \arg\min\{\mathcal{J}_n(\pi) : \pi$ obstacle-free in $\mathcal{G}(n)\}$
11:    $K(\bar{u}_n) \leftarrow \bar{\mathcal{J}}(\bar{\pi})$
12:  VISITED($\bar{u}_n$) $\leftarrow 1$
13:  if $\pi^*_n$ does not exist then
14:    if $\bar{u}_n = \bar{u}_{\mathrm{S}}$ then
15:      Report failure
16:    else
17:      $\bar{u}_{n+1} \leftarrow B(\bar{u}_n)$
18:  else
19:    $\bar{u}_{n+1} \leftarrow \mathsf{node}(\mathsf{cell}(u_1^{\pi^*_n};\mathcal{G}(n));\bar{\mathcal{G}})$
20:  $B(\bar{u}_n) \leftarrow \bar{u}_{n-1}$
21:  $J(\bar{u}_n) \leftarrow \mathcal{J}_n(\pi^*_n)$
22:  reachedGoal $\leftarrow (J(\bar{u}_n) = 0)$,
23:  $\bar{\pi} \leftarrow (\bar{\pi}, \bar{u}_n)$
24:  $\bar{\mathcal{J}}(\bar{\pi}) \leftarrow \bar{\mathcal{J}}(\bar{\pi}) + \bar{g}(\bar{u}_n, \bar{u}_{n+1})$
25:  $n \leftarrow n + 1$
26: if $\bar{\mathcal{J}}(\bar{\pi}) \geqslant M$ or $J(\bar{u}_n) \geqslant M$ then
27:   Report failure
```

Before proving the completeness of the preceding algorithm, we make a few comments regarding its execution.

Remark 1: The constrained optimization problem in Line 7 can be solved by an algorithm that finds the k shortest paths in a graph. Such algorithms have been reported, for instance, in [17]. We implicitly assume that the k shortest paths will be of strictly increasing costs. This assumption is not required for the algorithm's successful execution, but it enables a concise statement of the algorithm.

Remark 2: In [18], we describe in detail the procedure MR-GRAPH used in Line 4.

Remark 3: Due to Line 8, the cost of "back-tracking" is not added to the cumulative cost $\bar{\mathcal{J}}(\bar{\pi})$. Also, it follows from (3) and Line 21 that $J(\bar{u}) = 0$ if and only if $\bar{u} = \bar{u}_{\mathrm{G}}$.

We associate with each path $\pi_n(u_n, u_{\mathrm{G},n})$ in $\mathcal{G}(n)$ the set

$\mathcal{W}_n(\pi_n)$ defined by

$$\mathcal{W}_n(\pi_n) \overset{\text{def}}{=} \bigcup_{m=0}^{P(n)} W(u_m^{\pi_n}, V(n)). \quad (5)$$

The algorithm is said to *meet a setback* at iteration n if there exists no obstacle-free path $\pi_n(u_n, u_{\mathrm{G},n})$ in $\mathcal{G}(n)$ satisfying $\mathcal{W}_n(\pi_n) \subseteq \mathcal{W}_{n-1}(\pi^*_{n-1})$. We are now ready to state and prove the main result of this section.

Proposition 4: The proposed algorithm is complete: if there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}_\mathrm{S}$ to $\bar{u}_\mathrm{G}$, then the algorithm finds an obstacle-free path in a finite number of iterations. Otherwise, the algorithm reports failure after a finite number of iterations.

Proof: Note that because the set of nodes in $\bar{V}$ is finite, it follows by Proposition A.4 that the algorithm terminates after a finite number $N \in \mathbb{N}$ of iterations. To show completeness, first suppose that there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}_\mathrm{S}$ to $\bar{u}_\mathrm{G}$. We consider several cases.

First, suppose that the algorithm never visits any node in $\bar{V}$ more than once, and that the algorithm does not meet any setbacks. By Proposition A.3, $J(\bar{u}_{n-1}) - J(\bar{u}_n) \geqslant 1$ and the sequence $J(\bar{u}_n)$ decreases strictly monotonically. Since $J(\bar{u}_n) \geqslant 0$ for each $n \in \mathbb{N}$, and since $J(\bar{u}_1)$ is finite (by Corollary A.2), there exists $Q \leqslant N$, such that $J(\bar{u}_n) = 0$ for each $n \geqslant Q$. It follows by Line 22 that the algorithm terminates after Q iterations, and since $J(\bar{u}_Q) = 0$, the algorithm visits the goal node $\bar{u}_\mathrm{G}$ at iteration Q.

Next, suppose that the algorithm visits some nodes in $\bar{V}$ multiple times and that the algorithm never meets any setbacks. Note that the number of multiply visited nodes is finite because the algorithm terminates in a finite number of iterations. Then either of the following statements hold: (a) the algorithm terminates at iteration $Q \leqslant N$ such that $\bar{u}_Q$ is a multiply visited node, or (b) there exists $Q < N$ such that for each $n = Q+1, Q+2, \ldots$, the node $\bar{u}_n$ is visited exactly once by the algorithm. If Statement (a) holds, then $\bar{u}_Q \neq \bar{u}_\mathrm{G}$ due to by Lines 3 and 22, which in turn implies that the algorithm reports failure in Line 15. It follows by Line 14 that $\bar{u}_Q = \bar{u}_\mathrm{S}$. Then, by Proposition A.1 and Proposition A.5, there exists no admissible path in $\bar{\mathcal{G}}$ from $\bar{u}_\mathrm{S}$ to $\bar{u}_\mathrm{G}$, which is a contradiction. On the other hand, if Statement (b) holds, then by the monotonicity arguments in the preceding paragraph, the algorithm visits the goal in a finite number of iterations after iteration Q.

Next, suppose that the algorithm never visits any node in $\bar{V}$ more than once, and suppose that the algorithm meets some setbacks. The number of setbacks met by the algorithm is finite because the algorithm terminates in a finite number of iterations. Then either of the following statements hold: (c) the algorithm terminates at iteration $Q \leqslant N$ such that the algorithm meets a setback at iteration Q or (d) there exists $Q < N$ such that for each $n = Q+1, Q+2, \ldots$, such that the algorithm does not meet any setbacks after iteration Q. Statement (c) leads to the same contradiction that follows Statement (a), whereas Statement (d) leads to the same conclusion that follows Statement (b) above.

Finally, suppose that the algorithm visits some nodes in $\bar{V}$ multiple times and that algorithm meets some setbacks. We may combine the arguments in the two preceding paragraphs to either conclude that the algorithm visits the goal in a finite number of iterations, or to arrive at the contradiction that there exists no obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}_\mathrm{S}$ to $\bar{u}_\mathrm{G}$.

Now consider the case when there exists no obstacle-free path in the graph $\bar{\mathcal{G}}$ from the initial node $\bar{u}_\mathrm{S}$ to the goal node $\bar{u}_\mathrm{G}$. The set of nodes $\bar{V}$ is finite, hence it follows by Proposition A.4 that the algorithm terminates after a finite number of iterations. Suppose, for the sake of contradiction, that the algorithm erroneously finds a path $\bar{\pi}$ from the initial node $\bar{u}_\mathrm{S}$ to the goal node $\bar{u}_\mathrm{G}$. Then $\bar{\mathcal{J}}(\bar{\pi}) > M$, since $\bar{\pi}$ is not obstacle-free. It follows by Line 24 that $\bar{\mathcal{J}}(\bar{\pi}) > M$ at some intermediate iteration of the algorithm. However, by Line 3, the algorithm terminates whenever $\bar{\mathcal{J}}(\bar{\pi}) > M$, thus leading to a contradiction. Thus, the algorithm does not erroneously find a path from the node $\bar{u}_\mathrm{S}$ to the node $\bar{u}_\mathrm{G}$ if no obstacle-free path exists, and by Line 26, it reports failure in this case. ■

III. Efficient Constructions of the Graphs $\mathcal{G}(n)$

In this section, we describe a method to obtain $\mathcal{A}(n)$ efficiently by adding and removing elements from $\mathcal{A}(n-1)$. Specifically, we first determine the elements of the sets $\mathcal{B}_1 \overset{\text{def}}{=} \mathcal{A}(n) \cap \mathcal{A}^\mathrm{c}(n-1)$ and $\mathcal{B}_{-1} \overset{\text{def}}{=} \mathcal{A}(n-1) \cap \mathcal{A}^\mathrm{c}(n)$, and then evaluate $\mathcal{A}(n) = \mathcal{A}(n-1) \cup \mathcal{B}_1 \backslash \mathcal{B}_{-1}$. In light of the definition of $\mathcal{A}(n)$ in the procedure MR-Approx, we observe that $2^{j_{\max}} x(\bar{u}) = \lfloor 2^{j_{\max}} x(\bar{u}) \rfloor + \mathfrak{d}$ and $2^{j_{\max}} y(\bar{u}) = \lfloor 2^{j_{\max}} y(\bar{u}) \rfloor + \mathfrak{d}$, where $\mathfrak{d} \overset{\text{def}}{=} 2^{-j_{\max}-1}$ for $\bar{u} \in \bar{V}$. It can be shown that

$$\begin{aligned} \lfloor 2^j x(\bar{u}_{n+1}) \rfloor &= \lfloor \lfloor 2^j x(\bar{u}_n) \rfloor + 2^{j-j_{\max}} \Delta_x + r_x^j \rfloor, \quad (6) \\ \lfloor 2^j y(\bar{u}_{n+1}) \rfloor &= \lfloor \lfloor 2^j y(\bar{u}_n) \rfloor + 2^{j-j_{\max}} \Delta_y + r_y^j \rfloor, \quad (7) \end{aligned}$$

where $r_x^j \overset{\text{def}}{=} 2^{j-j_{\max}} \left(\lfloor 2^{j-j_{\max}} x(\bar{u}_n) \rfloor + \mathfrak{d} \right) - \lfloor 2^j x(\bar{u}_n) \rfloor$. The elements of the sets $\mathcal{B}_1$ and $\mathcal{B}_{-1}$ are then determined from (6)-(7) as follows. We define the scalar $\mathfrak{D}_x$ as

$$\mathfrak{D}_x \overset{\text{def}}{=} \begin{cases} -1, & 0 > 2^{j-j_{\max}} \Delta_x + r_x^j, \\ 0, & 0 \leqslant 2^{j-j_{\max}} \Delta_x + r_x^j < 1, \\ 1, & 1 \leqslant 2^{j-j_{\max}} \Delta_x + r_x^j, \end{cases} \quad (8)$$

and similarly for $\mathfrak{D}_y$. We then define the sets $\mathcal{B}_m^{j,x}$ by

$$\begin{aligned} \mathcal{B}_m^{j,x} \overset{\text{def}}{=} \big\{ (j,k,\ell) : &\, k = \lfloor 2^j x(\bar{u}_n) \rfloor + m \mathfrak{D}_x, \\ &\lfloor 2^j y(\bar{u}_n) \rfloor - \varrho(j) \leqslant \ell \leqslant \lfloor 2^j y(\bar{u}_n) \rfloor + \varrho(j) \big\}, \end{aligned}$$

where $m \in \{-1, 1\}$, and the sets $\mathcal{B}_m^{j,y}$, analogously. Then the sets $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ are given by the following relation

$$\mathcal{B}_m = \bigcup_{\alpha=\{x,y\}} \bigcup_{j_{\min} \leqslant j < j_{\max}} \mathcal{B}_m^{j,\alpha}, \quad m \in \{-1, 1\}. \quad (9)$$

The modified procedures for determining the elements of the set $\mathcal{A}(n)$ and the elements of the cell decomposition $\mathcal{C}^\mathrm{mr}(n)$ are then described as follows.

```
procedure Mod-MR-Approx($\mathcal{A}(n-1)$)
1: Compute $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ with (9)
2: $\mathcal{A}(n) \leftarrow \mathcal{A}(n) = \mathcal{A}(n-1) \cup \mathcal{B}_1 \backslash \mathcal{B}_{-1}$
```

```
procedure Mod-MR-Graph($\mathcal{C}^{\mathrm{mr}}(n-1), \mathcal{B}_{-1}, \mathcal{B}_1$)
1: $\mathcal{C}^{\mathrm{mr}}_{-1} \leftarrow \varnothing$, $\mathcal{C}^{\mathrm{mr}}_1 \leftarrow \varnothing$
```

```
2: for all $(j,k,\ell) \in \mathcal{B}_1$ do
3:     $\mathcal{C}_1^{\rm mr} \leftarrow \mathcal{C}_1^{\rm mr} \cup \{C(j+1,\hat{k},\hat{\ell}) : 2k \leqslant \hat{k} \leqslant 2k+1,\ 2\ell \leqslant \hat{\ell} \leqslant 2\ell+1\}$
4:     $\mathcal{C}_{-1}^{\rm mr} \leftarrow \mathcal{C}_{-1}^{\rm mr} \cup \{C(\hat{j},\hat{k},\hat{\ell}) : \hat{k} = \lfloor 2^{\hat{j}-j}k \rfloor,\ \hat{\ell} = \lfloor 2^{\hat{j}-j}\ell \rfloor,\ j_{\min} \leqslant \hat{j} \leqslant j\}$
5: for all $(j,k,\ell) \in \mathcal{B}_{-1}$ do
6:     $\mathcal{C}_{-1}^{\rm mr} \leftarrow \mathcal{C}_{-1}^{\rm mr} \cup \{C(j+1,\hat{k},\hat{\ell}) : 2k \leqslant \hat{k} \leqslant 2k+1,\ 2\ell \leqslant \hat{\ell} \leqslant 2\ell+1\}$
7:     $\mathcal{C}_1^{\rm mr} \leftarrow \mathcal{C}_1^{\rm mr} \cup \{C(j,k,\ell)\}$
8: $\mathcal{C}^{\rm mr}(n) \leftarrow \mathcal{C}^{\rm mr}(n-1) \cup \mathcal{C}_1^{\rm mr} \backslash \mathcal{C}_{-1}^{\rm mr}$
```

The advantage of computing $\mathcal{A}(n)$ using the modified procedure MOD-MR-APPROX instead of the procedure MR-APPROX arises from the fact that the number of elements in the set $\mathcal{A}(n)$ is $O(\bar{\varrho}^2)$, whereas the numbers of elements in the sets $\mathcal{B}_{-1}$ and $\mathcal{B}_1$ are both $O(\bar{\varrho})$, where $\bar{\varrho} \overset{\rm def}{=} \max_{j_{\min} \leqslant j \leqslant j_{\max}} \{\varrho(j)\}$. This observation also elicits the advantage of computing $\mathcal{C}^{\rm mr}(n)$ via procedure MOD-MR-GRAPH: the approach of directly computing[3] from $\mathcal{A}(n)$ executes in $O(\bar{\varrho}^2)$ time, because $O(\bar{\varrho}^2)$ iterations of the constant-time operations similar to those described in Lines 3-4 of procedure MOD-MR-GRAPH are performed. On the other hand, the procedure MOD-MR-GRAPH executes in $O(\bar{\varrho})$ time, because $O(\bar{\varrho})$ iterations of the constant-time operations in Lines 3-4 and Lines 6-7 are performed.

Remark 5: The graph $\mathcal{G}(n)$ is obtained from the graph $\mathcal{G}(n-1)$ by adding and deleting a relatively small number of nodes and edges. In light of this observation, the operation of finding the shortest path in $\mathcal{G}(n)$ may be performed using the so-called *incremental* algorithms [19], which reuse information about a previously known shortest path to find a new shortest path corresponding to changes in the graph.

Table I shows the results of evaluating through numerical simulations the ratio of the execution time required by the combination of the procedures MR-APPROX and MR-GRAPH to the execution time required by the combination of the procedures MOD-MR-APPROX and MOD-MR-GRAPH for computing the graph $\mathcal{G}(n)$. As predicted by the preceding theoretical analysis, the execution time ratios increase as the size $\bar{\varrho}$ of the high-resolution "window" increases. The execution time ratios also increase with $|\bar{V}| = 2^{2D}$, because we assume $j_{\min} = -D$. The execution time ratios in Table I were computed by averaging over 30 simulations, for each row of data in Table I.

Table II shows the results of evaluating through numerical simulations the ratio of the execution time of the entire path planning algorithm using procedures MR-APPROX and MR-GRAPH to the execution time of the entire path planning algorithm using procedures MOD-MR-APPROX and MOD-MR-GRAPH. The multi-resolution path planning algorithm with the modified procedures of construction of $\mathcal{A}(n)$ and $\mathcal{G}(n)$ executes up to 10 times faster.

[3]The elements of $\mathcal{C}^{\rm mr}(n)$ can be determined directly by Lines 2- 4 of the procedure MOD-MR-GRAPH after replacing $\mathcal{B}_1$ in Line 2 with $\mathcal{A}(n)$ and after appropriate initialization of $\mathcal{C}_1^{\rm mr}$; see Ref. [18] for details.

TABLE I
NUMERICAL COMPARISONS OF EXECUTION TIMES ILLUSTRATING EFFICIENT COMPUTATION OF $\mathcal{A}$ AND $\mathcal{G}$

2^D	$\bar{\varrho}$	Average exec. time ratio	2^D	$\bar{\varrho}$	Average exec. time ratio
128	4	8.0671	256	15	13.446
128	6	8.3114	256	30	21.135
128	15	12.552	512	15	18.886
256	4	9.9418	512	30	28.351

TABLE II
NUMERICAL COMPARISONS OF EXECUTION TIMES ILLUSTRATING BENEFITS IN OVERALL PATH PLANNING

2^D	$\bar{\varrho}$	Sample exec. time ratio	2^D	$\bar{\varrho}$	Sample exec. time ratio
128	4	6.7002	256	15	7.9145
128	6	10.163	512	4	8.0774
256	6	7.6615			

IV. PLANNING IN DYNAMIC ENVIRONMENTS

The path planning algorithm described in Section II and the modifications of the algorithm described in Section III assume a static environment, i.e., that the map F does not change. In this section, we describe an extension to the algorithm described in Section II that accounts for changes in the values of F.

In the context of the wavelet-based path planning algorithm, changes in the environment can be incorporated efficiently by updating the wavelet transform coefficients of F without recalculating *all* the coefficients. In what follows, we demonstrate a simple and efficient procedure for updating the wavelet transform coefficients.

Let F and $\tilde{F}$ denote, respectively, the original and the changed intensity maps. For the sake of simplicity, we assume that $\tilde{F}$ differs from F only in the value at (κ,λ), $\kappa,\lambda \in \{0,1,\ldots,2^{D-j_{\max}}-1\}$, i.e.,

$$\tilde{F}(k,\ell) = \begin{cases} F(k,\ell), & k \neq \kappa \text{ or } \ell \neq \lambda, \\ F(k,\ell)+\varepsilon, & k = \kappa \text{ and } \ell = \lambda, \end{cases}$$

where $\varepsilon \neq 0$ is a known scalar. For each $j_{\min} \leqslant j \leqslant j_{\max}$, $k,\ell = 0,1,\ldots,2^{D-j}$, we define a *scaled average intensity* $\mathfrak{F}$ by $\mathfrak{F}(j,k,\ell) \overset{\rm def}{=} \sum_{(\hat{k},\hat{\ell}) \in \mathcal{K}\times\mathcal{L}} F(\hat{k},\hat{\ell})$, where $\mathcal{K} \overset{\rm def}{=} \big[2^{-j}k, 2^{-j}(k+1)\big)$, and $\mathcal{L} \overset{\rm def}{=} \big[2^{-j}\ell, 2^{-j}(\ell+1)\big)$ We may similarly define the scaled average intensity $\tilde{\mathfrak{F}}$ corresponding to the intensity map $\tilde{F}$, and denote by $\epsilon^0_{j,k,\ell}$ the difference $\tilde{\mathfrak{F}}(j,k,\ell) - \mathfrak{F}(j,k,\ell)$, for $j_{\min} \leqslant j \leqslant j_{\max}$, $k,\ell = 0,1,\ldots,2^{D-j}$. Also, we denote by $\epsilon^i_{j,k,\ell}$ the difference $\tilde{d}^i_{j,k,\ell} - d^i_{j,k,\ell}$, for $i = 1,2,3$, where $\tilde{d}^i_{j,k,\ell}$ are the wavelet transform detail coefficients of the new intensity map $\tilde{F}$. It can be shown that

$$\begin{bmatrix} \epsilon^0_{j,k,\ell} \\ \epsilon^1_{j,k,\ell} \\ \epsilon^2_{j,k,\ell} \\ \epsilon^3_{j,k,\ell} \end{bmatrix} = \mathfrak{E} \begin{bmatrix} \epsilon^0_{j+1,\hat{k},\hat{\ell}} \\ \epsilon^0_{j+1,\hat{k},\hat{\ell}+1} \\ \epsilon^0_{j+1,\hat{k}+1,\hat{\ell}} \\ \epsilon^0_{j+1,\hat{k}+1,\hat{\ell}+1} \end{bmatrix}, \tag{10}$$

where $\mathfrak{E} \in \mathbb{R}^{4\times 4}$ is a constant, known matrix. For each $j_{\min} \leqslant j \leqslant j_{\max}$, $k \in \{2\lfloor 2^j\kappa \rfloor, 2\lfloor 2^j\kappa \rfloor + 1\}$, and $\ell \in$

$\{2\lfloor 2^j\lambda\rfloor, 2\lfloor 2^j\lambda\rfloor+1\}$ we may write

$$\epsilon^0_{j,k,\ell} = \begin{cases} 2^j\varepsilon, & k=\lfloor 2^{j+1}\kappa\rfloor \text{ and } \ell=\lfloor 2^{j+1}\lambda\rfloor \\ 0, & \text{otherwise.}\end{cases} \quad (11)$$

Equation (10) may be evaluated by substituting values in the right hand side using (11) to obtain the values of $\epsilon^i_{j,k,\ell}$, $(i=1,2,3)$, which are the differences between the wavelet transform detail coefficients of $\tilde{F}$ and F. Equation (11) may be used to directly evaluate the approximation coefficients $\tilde{a}_{j_{\min},k,\ell}$ corresponding to $\tilde{F}$, since $\tilde{a}_{j_{\min},k,\ell} = \tilde{\mathfrak{F}}(j_{\min},k,\ell)$ and $a_{j_{\min},k,\ell} = \mathfrak{F}(j_{\min},k,\ell)$, i.e., $\tilde{a}_{j_{\min},k,\ell} - a_{j_{\min},k,\ell} = \epsilon^0_{j_{\min},k,\ell}$.

The coefficient update schemes (10)-(11) are particularly beneficial when the number of cells for which the values of F changes is small compared to the total number of cells.

V. Conclusions and Future Work

In this paper, we have extended the results of [11], by proposing a modification of the original multi-resolution path planning algorithm, and have proved its completeness. In addition, we have presented computational procedures for efficient implementation of the algorithm, supported by numerical simulation results that illustrate the benefits of these procedures. Finally, we have presented a simple scheme for updating the wavelet transform coefficients of the map representing the environment (e.g., the obstacle space) stemming from small changes in the map. This scheme enables the application of the multi-resolution path planning algorithm to dynamic environments.

Acknowledgement: This research was supported in part by NASA (award no. NNX08AB94A).

References

[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[3] H. Samet, "The quadtree and related hierarchical data structures," *Computing Surveys*, vol. 16, no. 2, pp. 187–260, June 1984.

[4] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 3, pp. 135–45, September 1986.

[5] H. Noborio, T. Naniwa, and S. Arimoto, "A quadtree-based path planning algorithm for a mobile robot," *Journal of Robotic Systems*, vol. 7, no. 4, pp. 555–74, 1990.

[6] S. Behnke, "Local multiresolution path planning," *Lecture Notes in Artificial Intelligence*, vol. 3020, pp. 332–43, 2004.

[7] J. Y. Hwang, J. S. Kim, S. S. Lim, and K. H. Park, "A fast path planning by path graph optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 33, no. 1, pp. 121–127, January 2003.

[8] R. J. Prazenica, A. J. Kurdila, R. C. Sharpley, and J. Evers, "Multiresolution and adaptive path planning for maneuver of micro-air-vehicles in urban environments," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, San Francisco, CA, 2005, pp. 1–12.

[9] C.-T. Kim and J.-J. Lee, "Mobile robot navigation using multi-resolution electrostatic potential field," in *32nd Annual Conference of IEEE Industrial Electronics Society, 2005, IECON 2005*, 2005.

[10] B. J. H. Verwer, "A multiresolution workspace, multiresolution configuration space approach to solve the path planning problem," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, 1990, pp. 2107–12.

[11] P. Tsiotras and E. Bakolas, "A hierarchical on-line path planning scheme using wavelets," in *Proceedings of the European Control Conference*, Kos, Greece, July 2–5 2007, pp. 2806–2812.

[12] D. Jung, "Hierarchical path planning and control of a small fixed-wing UAV: Theory and experimental validation," Ph.D. dissertation, Georgia Institute of Technology, 2007.

[13] D. K. Pai and L.-M. Reissell, "Multiresolution rough terrain motion planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 19–33, February 1998.

[14] L. Carrioli, "Unsupervised path planning of many asynchronously self-moving vehicles," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91*, 1991, pp. 555–59.

[15] B. Sinopoli, M. Micheli, G. Donato, and T. J. Koo, "Vision based navigation for an unmanned aerial vehicle," in *Proceedings of 2001 IEEE Conference on Robotics and Automation*, 2001, pp. 1757–64.

[16] R. M. Rao and A. S. Bopardikar, *Wavelet Transforms - Introduction to Theory and Applications*. Addison-Wesley, 1998.

[17] U. Huckenbeck, *Extremal Paths in Graphs*. Berlin, Germany: Akademie Verlag, 1997.

[18] R. V. Cowlagi and P. Tsiotras, "Beyond quadtrees: Cell decompositions for path planning using the wavelet transform," in *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, LA, 12–14 Dec. 2007, pp. 1392–1397.

[19] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *Artificial Intelligence Magazine*, vol. 25, pp. 99–112, 2004.

Appendix

Proposition A.1: Let $\bar{u} \in \bar{V}$, and $\mathcal{A} = \text{MR-APPROX}(\bar{u})$. Let $\mathcal{C}^{\rm mr}$ and $\mathcal{G}=(V,E)$ be, respectively, the multi-resolution cell decomposition and the topological graph associated with $\mathcal{A}$. If there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}$ to $\bar{u}_{\rm G}$, then there exists an obstacle-free path in $\mathcal{G}$ from $u \overset{\rm def}{=} \mathsf{node}(\mathsf{cell}(\bar{u};\mathcal{C}^{\rm mr});\mathcal{G})$ to $u_{\rm G}$, where $u_{\rm G}\in V$ is the unique node that satisfies $\bar{u}_{\rm G} \in W(u_{\rm G}, V)$.

Proof: Let $\bar{\pi}(\bar{u},\bar{u}_{\rm G}) = (\bar{u}^{\bar{\pi}}_0, \bar{u}^{\bar{\pi}}_1, \ldots, \bar{u}^{\bar{\pi}}_{\bar{P}})$ be an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}^{\bar{\pi}}_0 = \bar{u}$ to $\bar{u}^{\bar{\pi}}_{\bar{P}} = \bar{u}_{\rm G}$. For each $m=0,1,\ldots,\bar{P}$, there exists a unique $U_m \in \{W(u,V)\}_{u\in V}$ such that $\bar{u}^{\bar{\pi}}_m \in U_m$. Let $w_m \in V$ be such that $U_m = W(w_m, V)$. Because $\bar{\pi}$ is a path in $\bar{\mathcal{G}}$, $(\bar{u}^{\bar{\pi}}_{m-1}, \bar{u}^{\bar{\pi}}_m) \in \bar{E}$ for each $m=1,2,\ldots,\bar{P}$, and it follows that either $U_{m-1}=U_m$, or $(w_{m-1},w_m)\in E$. Thus, the path $\pi(u,u_{\rm G}) \overset{\rm def}{=} \{u^{\pi}_0, u^{\pi}_1, \ldots, u^{\pi}_P\}$, where $P \le \bar{P}$, is a path in $\mathcal{G}$.

To show that the path π is also obstacle-free in $\mathcal{G}$, we note that since $\bar{\pi}$ is obstacle-free in $\bar{\mathcal{G}}$, $F(\bar{u}^{\bar{\pi}}_m)=0$, for all $m=0,1,\ldots,\bar{P}$. It follows from (1) that $F^{\rm mr}(\mathsf{cell}(u^{\pi}_m;\mathcal{C}^{\rm mr})) < M$ for each $m=0,1,\ldots,P$, and from (3)-(4) that $\mathcal{J}(\pi) < M$, i.e., π is an obstacle-free path. ■

Corollary A.2: If there exists an obstacle-free path in $\bar{\mathcal{G}}$ from the initial node $\bar{u}_{\rm S}$ to the goal node $\bar{u}_{\rm G}$, then the cost of the initial path π^*_0 computed by the algorithm is finite.

Proof: By Proposition A.1, if there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}$ to $\bar{u}_{\rm G}$, then there exists an obstacle-free path $\pi^*_0(u_{\rm S}, u_{\rm G,0})$ in $\mathcal{G}(0)$ from the node $u_{\rm S} \overset{\rm def}{=} \mathsf{node}(\mathsf{cell}(\bar{u}_{\rm S};\mathcal{C}_{j_{\max}});\mathcal{G}(0))$ to the node $u_{\rm G,0}$, where $u_{\rm G,0} \in V(0)$ is the unique node that satisfies $\bar{u}_{\rm G} \in W(u_{\rm G,0}, V(0))$. Because π^*_0 is obstacle-free, $\mathcal{J}_0(\pi^*_0) < M$, i.e., $\mathcal{J}_0(\pi^*_0)$ is finite. ■

Proposition A.3: Suppose that the algorithm does not meet a setback at iteration $n \in \mathbb{N}$ of its execution, and also suppose that $\text{VISITED}(\bar{u}_n) = 0$. If there exists a path in the graph $\mathcal{G}(n)$ from the node $u_n = \mathsf{node}(\mathsf{cell}(\bar{u}_n;\mathcal{C}^{\rm mr}(n));\mathcal{G}(n))$ to the node $u_{{\rm G},n}$, then $J(\bar{u}_{n-1}) - J(\bar{u}_n) \geqslant 1$, where $u_{{\rm G},n} \in V(n)$ is the unique node that satisfies $\bar{u}_{\rm G} \in W(u_{{\rm G},n}, V(n))$.

Proof: Let $\pi^*_n(u_n, u_{{\rm G},n}) = (u^{\pi^*_n}_0, u^{\pi^*_n}_1, \ldots, u^{\pi^*_n}_{P(n)})$ denote the optimal path in the graph $\mathcal{G}(n)$ computed by the algorithm at Line 10. First, suppose that the cell decomposition $\mathcal{C}^{\mathrm{mr}}(n)$ is identical to the cell decomposition $\mathcal{C}^{\mathrm{mr}}(n-1)$ (in particular, $u_{\mathrm{G},n-1} = u_{\mathrm{G},n}$). If there exists a path in $\mathcal{G}(n)$ from u_n to $u_{\mathrm{G},n}$, then there exists an optimal path in $\mathcal{G}(n)$ from u_n to $u_{\mathrm{G},n}$ because $\mathcal{G}(n)$ is finite. Then, by Bellman's principle of optimality, the path $\pi^*_{n-1}(u_{n-1}, u_{\mathrm{G},n-1}) = (u_0^{\pi^*_{n-1}}, u_1^{\pi^*_{n-1}}, \ldots, u_{P(n-1)}^{\pi^*_{n-1}})$, computed at iteration $n-1$ of the algorithm, contains the path π^*_n, with $P(n) = P(n-1) - 1$, and $u_{m-1}^{\pi^*_n} = u_m^{\pi^*_{n-1}}$ for each $m = 1, 2, \ldots, P(n)$, and hence $\mathcal{J}_n(\pi^*_n) \leqslant \mathcal{J}_{n-1}(\pi^*_{n-1})$.

Next, suppose that the cell decomposition $\mathcal{C}^{\mathrm{mr}}(n)$ is not identical to the cell decomposition $\mathcal{C}^{\mathrm{mr}}(n-1)$. Let $\pi_n(u_n, u_{\mathrm{G},n})$ and $\pi_{n-1}(u_{n-1}, u_{\mathrm{G},n-1})$ be paths in the graphs $\mathcal{G}(n)$ and $\mathcal{G}(n-1)$ respectively. If $\mathcal{W}_n(\pi_n) \subseteq \mathcal{W}_{n-1}(\pi_{n-1})$, then due to the second term in the right hand side of (4), $\mathcal{J}_n(\pi_n) \leqslant \mathcal{J}_{n-1}(\pi_{n-1})$. In particular, if $\mathcal{W}_n(\pi^*_n) \subseteq \mathcal{W}_{n-1}(\pi^*_{n-1})$, then $\mathcal{J}_n(\pi^*_n) \leqslant \mathcal{J}_{n-1}(\pi^*_{n-1})$.

Now suppose $\mathcal{W}_n(\pi^*_n) \nsubseteq \mathcal{W}_{n-1}(\pi^*_{n-1})$. Let $\pi_n(u_n, u_{\mathrm{G},n})$ be any path in $\mathcal{G}(n)$ from u_n to $u_{\mathrm{G},n}$ satisfying $\mathcal{W}_n(\pi_n) \subseteq \mathcal{W}_{n-1}(\pi^*_{n-1})$. There exists at least one such path π_n in $\mathcal{G}(n)$ because the algorithm does not meet a setback at iteration n. By the arguments in the preceding paragraph, $\mathcal{J}_n(\pi_n) \leqslant \mathcal{J}_{n-1}(\pi^*_{n-1})$. Furthermore, because π^*_n is an optimal path in $\mathcal{G}(n)$ from u_n to $u_{\mathrm{G},n}$, $\mathcal{J}_n(\pi^*_n) \leqslant \mathcal{J}_n(\pi_n)$, and it follows that $\mathcal{J}_n(\pi^*_n) \leqslant \mathcal{J}_{n-1}(\pi^*_{n-1})$.

Finally, note that the cell corresponding to the first node $u_0^{\pi^*_n} \in V(n)$ in the path π^*_n is the same as the cell corresponding to the second node $u_1^{\pi^*_{n-1}} \in V(n-1)$ in π^*_{n-1}, and furthermore, this cell corresponds to the node $\bar{u}_n \in \bar{V}$. Then $J(\bar{u}_{n-1}) - J(\bar{u}_n) = \mathcal{J}_{n-1}(\pi^*_{n-1}) - \mathcal{J}_n(\pi^*_n) \geqslant \bar{g}(\bar{u}_{n-1}, \bar{u}_n) = 1$, by (2). ■

Proposition A.4: Let $\bar{u}$ be an arbitrary node in $\bar{V}$. Then either the algorithm never visits $\bar{u}$ or the algorithm visits $\bar{u}$ finitely many times.

Proof: Suppose, for the sake of contradiction, that the algorithm visits the node $\bar{u} \in \bar{V}$ infinitely many times at iterations $n_1, n_2, \ldots, n_k \ldots$, i.e., $\bar{u}_{n_1} = \bar{u}_{n_2} = \ldots = \bar{u}$. By Line 7, $J(\bar{u}_{n_k}) - J(\bar{u}_{n_{k-1}}) \geqslant 1$, and hence there exists $N \in \mathbb{N}$, such that $J(\bar{u}_{n_N}) \geqslant M$. It follows by Line 3 that the algorithm terminates in at most n_N iterations, leading to a contradiction. ■

Proposition A.5: Let $\pi^*_n(u_n, u_{\mathrm{G},n}) = (u_0^{\pi^*_n}, \ldots, u_{P(n)}^{\pi^*_n})$ be the path found by the algorithm either at Line 7 or Line 10 at iteration $n \in \mathbb{N}$, and suppose there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}_n$ to $\bar{u}_{\mathrm{G}}$ that is contained within the set $\mathcal{W}_n(\pi^*_n)$. Then the algorithm does not visit the node $\bar{u}_n$ at any future iteration.

Proof: Suppose, for the sake of contradiction, that there exists $\ell > 1$ such that the algorithm visits node $\bar{u}_n$ again at iteration $n+\ell$, i.e., $\bar{u}_n = \bar{u}_{n+\ell}$ and $\bar{u}_{n+1} = \bar{u}_{n+\ell-1}$. Then, due to the definition of admissible paths in the graphs $\mathcal{G}(n+k)$, $k \in \mathbb{N}$, which precludes cycles, there exists $m < \ell$ such that for each $k = m, m+1, \ldots, \ell$, the algorithm executes Line 17 at iteration $n+k$, i.e. $\bar{u}_{n+k+1} = B(\bar{u}_{n+k})$. In particular, for $k = \ell - 1$, $\bar{u}_{n+\ell} = \bar{u}_n = B(\bar{u}_{n+\ell-1}) = B(\bar{u}_{n+1})$. Note that, by Lines 17 and 13 and by Proposition A.1, $\bar{u}_{n+\ell} = B(\bar{u}_{n+\ell-1})$ implies either that there exists no path in $\bar{\mathcal{G}}$ from $\bar{u}_{n+\ell-1} = \bar{u}_{n+1}$ to $\bar{u}_{\mathrm{G}}$, or that every obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}_{n+\ell-1}$ to $\bar{u}_{\mathrm{G}}$ contains $B(\bar{u}_{n+\ell-1}) = \bar{u}_n$. Recall now that the cell corresponding to the second node in the path π^*_n is a cell at the finest resolution $j_{\max}$, and hence, $W(u_1^{\pi^*_n}, V(n)) = \bar{u}_{n+1}$. Then, by (5) and by the stated hypothesis, it follows that there exists an obstacle-free path $\bar{\pi}(\bar{u}_n, \bar{u}_{\mathrm{G}}) = (\bar{u}_0^{\bar{\pi}}, \ldots, \bar{u}_{\bar{P}}^{\bar{\pi}})$ in $\bar{\mathcal{G}}$ from $\bar{u}_n$ to $\bar{u}_{\mathrm{G}}$ such that $\bar{u}_1^{\bar{\pi}} = \bar{u}_{n+1}$. Thus, there exists an obstacle-free path in $\bar{\mathcal{G}}$ from $\bar{u}_{n+1}$ to $\bar{u}_{\mathrm{G}}$ that does not contain $\bar{u}_n$: in particular, $(\bar{u}_1^{\bar{\pi}}, \ldots, \bar{u}_{\bar{P}}^{\bar{\pi}})$ is such a path. The implication of the preceding paragraph contradicts this observation, and hence, the supposition that there exists $\ell > 1$ such that $\bar{u}_n = \bar{u}_{n+\ell}$ is false, i.e., the algorithm does not visit the node $\bar{u}_n$ at any future iteration. ■