Multiresolution On-Line Path Planning using Wavelet for a Small Unmanned Air Vehicle

Dongwon Jung, *Student Member, IEEE* and Panagiotis Tsiotras, *Senior Member, IEEE*

Abstract

In this article we propose a new online multiresolution path planning algorithm for a small unmanned air vehicle (UAV) with limited on-board computational resources. The proposed approach assumes that the UAV has detailed information of the environment and the obstacles only in its vicinity. Information for far away obstacles is also available, albeit less accurately. The proposed algorithm uses the fast lifting wavelet transform (FLWT) to get a multiresolution cell decomposition of the environment, whose dimension is commensurate to the on-board computational resources. A topological graph representation of the multiresolution cell decomposition is constructed efficiently directly from the approximation and detail wavelet coefficients. A dynamic path planning is sequentially executed for an optimal path by using the $\mathcal{A}^*$ algorithm over the ensuing graph. The proposed path planning algorithm is implemented on-line on a small autopilot. Hardware-in-the-loop simulation (HILS) results validate the applicability of the algorithm on the actual system. Comparisons with the standard $\mathcal{D}^*$-lite algorithm are also presented.

I. Introduction

Autonomous operation of UAVs requires both trajectory design (planning) and trajectory tracking (control) tasks to be completely automated. Given the short response time scales of modern aerial vehicles, these are challenging tasks using existing route optimizers. On-board, real-time path planning is particulary challenging for small UAVs, which may not have the on-board computational capabilities (e.g. CPU and memory) to implement some of the sophisticated path planning algorithms proposed in the literature. In most applications this problem is bypassed by providing navigation way-points that have been computed either off-line, or on-line by a more capable supervising/leader agent.

In a typical mission of a UAV, various sensors (e.g., cameras, radars, laser scanners, satellite imagery) having different range and resolution characteristics are employed to collect information about the environment the vehicle operates in. A computationally efficient path planning algorithm, specifically adopted for on-line implementation, should therefore choose the expedient information from all these sensors, and use the on-board computational resources to design the part of the path (spatial and temporal) that needs it most. In a nutshell, a computationally efficient algorithm suitable for *on-line* implementation should be characterized by a combination of short term tactics (reaction to unforeseen threats) with long-term strategy (planning towards the ultimate goal).

Several multiresolution or hierarchical path planning algorithms have been proposed in the literature to alleviate the computational burden associated with path planning over a complex, unstructured, or partially known environment [1]–[5]. Quadtree decompositions have been used to get a decomposition of the environment for path planning purposes [6]–[9]. One drawback of quadtree-based decompositions is that a finer resolution is used close to the boundaries of all obstacles, regardless of their distance from the agent. This tends to waste computational resources.

Recently, Tsiotras and Bakolas [10] proposed an efficient hierarchical path planning algorithm for autonomous agents navigating in a partially known environment $\mathcal{W}$ using an adaptive, discrete, cell-based approximation of $\mathcal{W}$. The innovation of their approach hinges on the use of distinct levels of fidelity (resolution) of $\mathcal{W}$ at different distances from the agent's current position. A high resolution representation of $\mathcal{W}$ is used close to the current position of the agent (leading to a local solution with great accuracy), while a low resolution representation is used far away from the vehicle (thus incorporating the ultimate goal objective).

In this article, we assume a world environment $\mathcal{W} \subset \mathbb{R}^2$ that includes the obstacle space $\mathcal{O} \subset \mathcal{W}$ and the obstacle-free configuration space $\mathcal{F} = \mathcal{W} \setminus \mathcal{O}$ of all feasible states. We employ the wavelet transform to perform the required

D. Jung is a graduate research assistant, e-mail: dongwon_jung@ae.gatech.edu
P. Tsiotras is with the School of Aerospace, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: tsiotras@gatech.edu

multiresolution decomposition of $\mathcal{W}$. The fast lifting wavelet transform (FLWT) offers a fast decomposition of a function at different levels of resolution, which can be twice as fast as the classical wavelet transform[1]. Furthermore, the FLWT can be implemented using the integer arithmetic, which reduces the computational cost dramatically. This makes the FLWT especially suitable for processing using small micro-controllers. The use of FLWT has also the added benefit of allowing the construction of the associated cell connectivity relationship directly from the wavelet coefficients, thus eliminating the need for quadtree decomposition, as in Ref. [10].

We employ the hierarchical path planning principle to find the optimal path on the topological graph $\mathcal{G}$ induced by the previous wavelet-based cell decomposition. Namely, the optimal path from an initial state to a final state may contain mixed nodes across all resolution levels except at the finer resolution level, where the path is resolved through feasible states only. Hierarchical path planning is known to be more flexible than other methods that search only through free nodes [12].

In the sequel, we present a multiresolution hierarchical path planning algorithm, which is an extension of the algorithm developed in Ref. [10], and deals with the connectivity relationship between cells of varying sizes. The paper is organized as follows. In Section II we describe a multiresolution decomposition of $\mathcal{W}$ using the 2D Haar wavelet system. In Section III we present an efficient algorithm for constructing the adjacency list of a topological graph by the direct use of the wavelet coefficients. The multiresolution hierarchical path planning algorithm is shown in Section IV. Based on the hardware in-the-loop simulation (HILS) results in Section V, we discuss the advantages and disadvantages of the proposed algorithm over a standard dynamic, incremental path planning algorithm used in the literature.

II. A Multiresolution Decomposition of $\mathcal{W}$

A. The 2D wavelet transform

The idea behind the wavelet transform is to represent a function $f \in \mathcal{L}^2(\mathbb{R})$ via a linear combination of elementary basis functions $\phi_{J,k}$ and $\psi_{j,k}$ as follows

$$f(x) = \sum_{k\in\mathbb{Z}} a_{J,k}\phi_{J,k}(x) + \sum_{j\geq J}\sum_{k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(x), \tag{1}$$

where $\phi_{J,k}(x) = 2^{J/2}\phi(2^J x - k)$ and $\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$. The choice of J determines the low resolution, or the coarse approximation of f, spanned by the scaling function $\phi_{J,k}(x)$. The rest of $\mathcal{L}^2(\mathbb{R})$ is spanned by the wavelet functions $\psi_{j,k}(x)$ which provide the higher, or finer resolution details of the function. In other words, when analyzing the function f at the coarsest level (low resolution), only the most salient features of f will be revealed. Adding finer levels (high resolution) implies adding more and more details of the function f. The expansion (1) thus reveals the properties of f at different levels of resolution [13], [14]. In addition, in the ideal case both the scaling function and the wavelet function have compact support, that is they are non-zero only on a finite interval so they can capture the localized features of f.

The wavelet transform can be readily extended to the two-dimensional case by introducing the following families of functions

$$\Phi_{j,k,\ell}(x,y) = \phi_{j,k}(x)\phi_{j,\ell}(y), \tag{2a}$$
$$\Psi^1_{j,k,\ell}(x,y) = \phi_{j,k}(x)\psi_{j,\ell}(y), \tag{2b}$$
$$\Psi^2_{j,k,\ell}(x,y) = \psi_{j,k}(x)\phi_{j,\ell}(y), \tag{2c}$$
$$\Psi^3_{j,k,\ell}(x,y) = \psi_{j,k}(x)\psi_{j,\ell}(y). \tag{2d}$$

Given a function $f \in \mathcal{L}^2(\mathbb{R}^2)$ we can then write

$$f(x,y) = \sum_{k,\ell\in\mathbb{Z}} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j\geq J}\sum_{k,\ell\in\mathbb{Z}} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y) \tag{3}$$

[1]The computational complexity of the lifting scheme is still of order $\mathcal{O}(n)$ where n is the input data [11], however, the computational time may decrease by half according to wavelet basis.

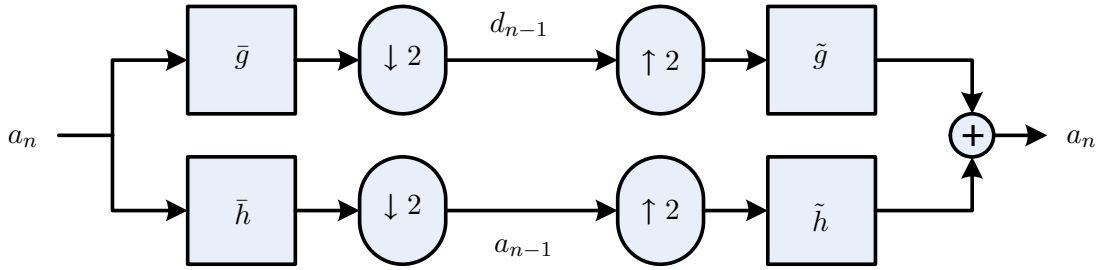


Fig. 1. A typical one-stage two-band filter banks used for implementing the discrete wavelet transform.

where, for the case of orthonormal wavelets, the approximation coefficients are given by[2]

$$a_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Phi_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{4}$$

and the detail coefficients by

$$d^i_{j,k,\ell} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\Psi^i_{j,k,\ell}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{5}$$

The key property of wavelets used in this paper is the fact that the expansion (3) induces the following multiresolution decomposition of $\mathcal{L}^2(\mathbb{R}^2)$

$$\mathcal{L}^2(\mathbb{R}^2) = \mathcal{V}_J \oplus \mathcal{W}_J \oplus \mathcal{W}_{J+1} \oplus \cdots, \tag{6}$$

where $\mathcal{V}_J = \overline{\mathrm{span}_{k,\ell\in\mathbb{Z}}\{\Phi_{J,k,\ell}\}}$ and $\mathcal{W}_j = \overline{\mathrm{span}_{k,\ell\in\mathbb{Z}}\{\Psi^1_{j,k,\ell}, \Psi^2_{j,k,\ell}, \Psi^3_{j,k,\ell}\}}$ for $j \geq J$.

In this paper we use the Haar wavelet system for reasons that will become apparent shortly. The Haar scaling function

$$\phi(x) = \begin{cases} 1 & \text{if } x \in [0,1), \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

and the Haar wavelet function

$$\psi(x) = \begin{cases} 1 & \text{if } x \in [0,1/2), \\ -1 & \text{if } x \in [1/2,1), \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

have compact support on $[0,1]$. Hence, each scaling function $\phi_{j,k}(x)$ and wavelet function $\psi_{j,k}(x)$ in the Haar system has support on the dyadic interval $I_{j,k} \triangleq [k/2^j, (k+1)/2^j]$ of length $1/2^j$ and does not vanish in this interval [13], [15]. Subsequently, we may associate the two-dimensional scaling function $\Phi_{j,k,\ell}$ and the wavelet function $\Psi^i_{j,k,\ell}$ $(i = 1,2,3)$ with the rectangular cell $c^j_{k,\ell} \triangleq I_{j,k} \times I_{j,\ell}$.

B. Fast lifting wavelet transform (FLWT)

Implementing the wavelet transform in practice requires dealing with a discrete signal. The basic step in a typical discrete wavelet transform (DWT) involves the use of filter banks. Figure 1 shows a discrete signal a_n filtered by two complementary high- and low-pass (decomposition) filters $\bar{g}$ and $\bar{h}$ before it is down-sampled. The results of this operation are the next coarser approximation and detail coefficients a_{n-1} and d_{n-1}, each containing half as many samples as the input signal a_n. For the inverse transform, first the signals a_{n-1} and d_{n-1} are upsampled by inserting zeroes between every sample. Subsequently, the two signals are filtered by the low- and high-pass (reconstruction) filters $\tilde{g}$ and $\tilde{h}$, respectively, and then added together. This sequence of operations results in perfect reconstruction of the original signal a_n. Details of the filter bank implementation of wavelet transforms can be found, for instance, in Refs. [16] and [17].

[2]In the more general case of biorthogonal wavelets projections on the space spanned by the dual wavelets and dual scaling functions should be used in (4) and (5).

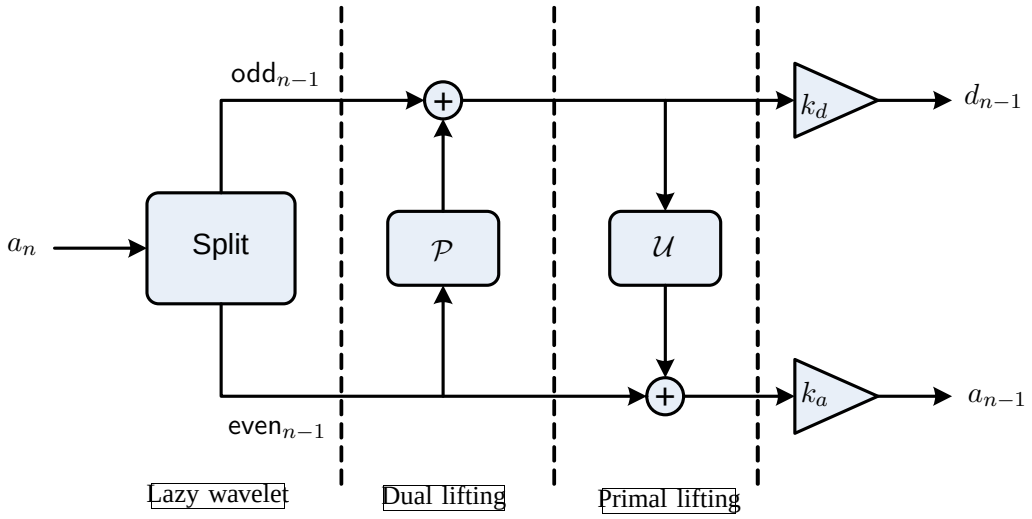


Fig. 2. One step decomposition using the lifting scheme with the lazy wavelet.

The fast lifting wavelet scheme, originally introduced in Refs. [18] and [19], is a new method for building wavelets directly in the time domain, thus avoiding the use of Fourier analysis. Moreover, the scheme can be extended to construct the so-called second generation wavelets, which have certain benefits for handling boundary effects, irregular samples, and arbitrary weight functions [17].

The typical lifting decomposition scheme is depicted in Fig. 2. The first block in this decomposition splits the original signal a_n into two disjoint sets of samples containing the odd and the even indexed samples (Lazy wavelet). Because the even and odd subsets are correlated to each other locally, each signal is lifted by the opposite signal after passing through the corresponding operators $\mathcal{P}$ and $\mathcal{U}$ (the dual and primal lifting, or the prediction and update, respectively). Finally, the results are normalized with the constants k_a and k_d, to end up with the approximation and detail coefficients, a_{n-1} and d_{n-1}, respectively.

For the case of the unnormalized Haar transform, the dual lifting does nothing more but calculate the difference of two signals

$$d_{n-1,k} = a_{n,2k+1} - a_{n,2k}, \tag{9}$$

whereas the primal lifting calculates the coarser approximation coefficients having the same average value as the original signal, by updating the even samples using the previously calculated detail signal as follows

$$a_{n-1,k} = a_{n,2k} + d_{n-1,k}/2. \tag{10}$$

It has been proved that all classical wavelet transforms can be implemented using the lifting scheme [20]. Most interestingly, the inverse transform is readily found by reversing the order of the operations and by flipping the signs.

The lifting scheme has a number of algorithmic advantages, such as faster computation speed (twice as fast as the usual discrete wavelet transform), *in-place* calculation of the coefficients (that saves memory), immediate inverse transform, generality for extension to irregular problems, etc. In particular, the lifting scheme is applicable to many applications where the input data consists of integer samples. Unlike the typical wavelet transform where floating number arithmetic is implicitly assumed, the lifting scheme can be easily modified to map integers to integers, and is readily reversible to allow perfect reconstruction [21]. This reconstruction is possible by adopting the sequential transform by modifying Eq. (10) as follows [22]

$$\begin{aligned} d_{n-1,k} &= a_{n,2k+1} - a_{n,2k}, \\ a_{n-1,k} &= a_{n,2k} + \lfloor d_{n-1,k}/2 \rfloor, \end{aligned} \tag{11}$$

where, $\lfloor \cdot \rfloor$ is the rounding operator. In the sequel, we use the fast lifting Haar transform for two-dimensional signals of integer samples using a sequential 2D scheme; that is, we perform two one-dimensional transforms through the rows and then columns of the input data.

C. Wavelet decomposition of the risk measure

Without loss of generality, we let $\mathcal{W} = [0,\ 1] \times [0,\ 1]$, which is described using a discrete (fine) grid of $2^N \times 2^N$ dyadic points. The finest level of resolution $J_{\max}$ is therefore bounded by N. It follows from Eq. (3) and the accompanying discussion that the Haar wavelet decomposition at resolution level $J \geq J_{\min}$, given by

$$f(x,y) = \sum_{k,\ell=0}^{2^J-1} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J}^{N-1}\sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y), \tag{12}$$

and it induces a cell decomposition of $\mathcal{W}$ of square cells of maximum size $1/2^J \times 1/2^J$.

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto \mathcal{M}$ that represents the "risk measure" at the location $\mathsf{x} = (x, y)$, where $\mathcal{M}$ is a collection of m integer distinct risk measure levels defined by

$$\mathcal{M} \triangleq \{M_i : M_1 < M_2 < \cdots < M_m\}. \tag{13}$$

The obstacle space $\mathcal{O}$ is defined as the space where the risk measure values exceed a certain threshold $\overline{M}$, that is,

$$\mathcal{O} = \{\mathsf{x} \in \mathcal{W} \mid \mathsf{rm}(\mathsf{x}) > \overline{M},\ \overline{M} \in \mathcal{M}\}. \tag{14}$$

For $\mathsf{x} \in \mathcal{F}$, we may think of $\mathsf{rm}(\mathsf{x})$ as an indication of the proximity of the agent to the obstacle space or the probability that $\mathsf{x} \in \mathcal{O}$.

We construct approximations of $\mathcal{W}$ at distinct levels of resolution $J_{\min} \leq j \leq J_{\max}$ at ranges r_j from the current location of the agent $\mathsf{x}_0 = (x_0, y_0)$, in the sense that the resolution j is used for all points inside the neighborhood

$$\mathcal{N}(\mathsf{x}_0, r_j) \triangleq \{\mathsf{x} \in \mathcal{W} : \|\mathsf{x} - \mathsf{x}_0\|_\infty \leq r_j\}. \tag{15}$$

where $r_{J_{\max}} \leq r_j \leq r_{J_{\min}}$. By this, we imply that the finer resolution $J_{\max}$ is used for points close to the current location, and coarser resolutions at different levels are used elsewhere, according to the distance from the current location. Hence, the representation of $\mathcal{W}$ gets coarser further away from the current location. Figure 3 illustrates this situation. The choice of $J_{\max}$ is determined by the requirement that at this level all cells can be resolved into either free or obstacle cells. The choice of $J_{\min}$ as well as the window span r_j are dictated by the on-board computational resources.

Let now $\mathcal{I}(j) \triangleq \{0, 1, 3, \cdots, 2^j - 1\}$ and let

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) \mid I_{j,k} \cap [x_0 - r_j, x_0 + r_j] \neq \varnothing\}, \tag{16a}$$

$$\mathcal{L}(j) \triangleq \{\ell \in \mathcal{I}(j) \mid I_{j,\ell} \cap [y_0 - r_j, y_0 + r_j] \neq \varnothing\}. \tag{16b}$$

Then the wavelet decomposition of rm, given by

$$\mathsf{rm}(x,y) = \sum_{k,\ell \in \mathcal{I}(J_{\min})} a_{J_{\min},k,\ell}\Phi_{J_{\min},k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J_{\min}}^{J_{\max}-1}\sum_{\substack{k\in\mathcal{K}(j)\\ \ell\in\mathcal{L}(j)}} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y), \tag{17}$$

induces, with a slight abuse of notation, the following multiresolution cell decomposition on $\mathcal{W}$

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}}, \tag{18}$$

where, ΔC_d^j is a union of cells $c^j_{k,\ell}$ of dimension $1/2^j \times 1/2^j$.

III. Multiresolution Graph Connectivity

A. Computation of Adjacency List from the FLWT

In the previous section we described the construction of a multiresolution cell decomposition $\mathcal{C}_d$ in (18) of $\mathcal{W}$ using the FLWT. We now assign a topological graph $\mathcal{G} = (V, E)$ to $\mathcal{C}_d$ as follows. The nodes of $\mathcal{G}$ represent the cells $c^j_{k,\ell}$ in $\mathcal{C}_d$ and the edges represent the connectivity relationship between those nodes. In this section we show that the connectivity of the graph $\mathcal{G}$ can be constructed directly from the wavelet coefficients. Equivalently, we compute the adjacency list of $\mathcal{G}$ directly from wavelet coefficients obtained from the FLWT.

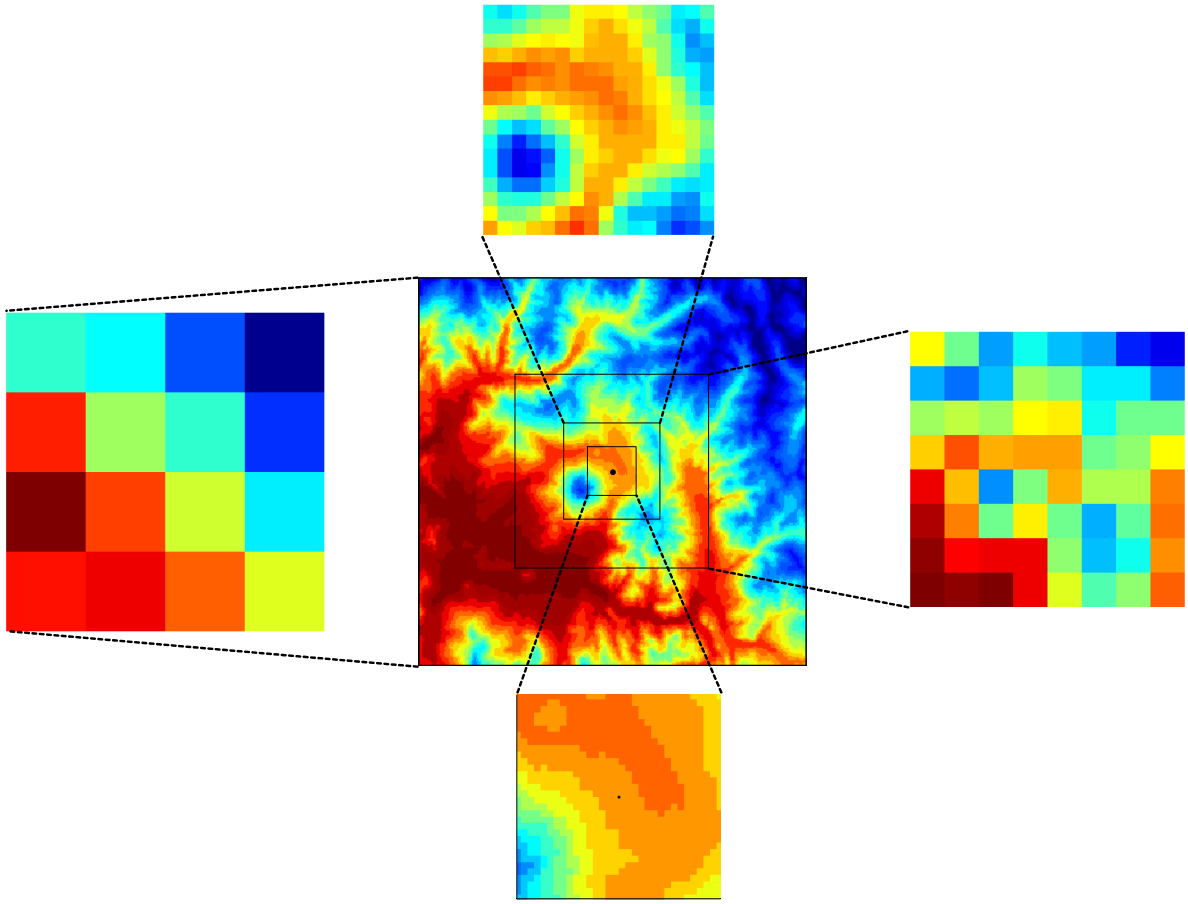


Fig. 3. Multiresolution representation of the environment according to the distance from the current location of the agent.

As the scaling function $\Phi_{j,k,\ell}$ and the wavelet functions $\Psi^i_{j,k,\ell}$ $(i = 1, 2, 3)$ of the 2D Haar wavelet are associated with square cells $c^j_{k,\ell}$, the corresponding approximation and nonzero detail coefficients encode the necessary information regarding the cell geometry (size and location). Recall that the approximation coefficients are the average values of the risk measure values over the cells, and the detail coefficients determine the size of each cell. To this end, consider a cell $c^{j_0}_{k,\ell}$ at level j_0, whose dimension is $1/2^{j_0} \times 1/2^{j_0}$ and is located at (k, ℓ). A cell will be called *independent* if it is associated with a non-zero approximation coefficient $a_{j_0,k,\ell}$, while the corresponding detail coefficients $d^i_{j,k,\ell}$ $(i = 1, 2, 3)$ at level $j_0 \leq j \leq J_{\max}$ are all zero. Otherwise, the cell is marked as a *parent* cell, and is subdivided into four *leaf* cells at level $j_0 + 1$. If a leaf cell cannot be subdivided further, it is classified as an independent cell. In Fig. 4, the top-most parent cell $c^{j_0}_{k,\ell}$ is subdivided into three independent cells at level $j_0 + 1$ with each non-zero approximation coefficient in the quadrant I, II, and III (all zero detail coefficients). For quadrant IV, the cell is further subdivided into four independent leaf cells at level $j_0 + 2$.

Assume now that we are given the Haar wavelet transform of the risk measure function rm up to the level $J_{\min}$. The coarsest level of the cell dimension is set to $J_{\min}$. In Fig. 5 the initial coarse grid is drawn on the left. The agent is located at $\mathsf{x} = (x, y)$ and the high resolution horizon is given by r. Recalling expressions (15), we distinguish cells at distinct resolution levels, by starting from a coarse cell $c^{j_0}_{k,\ell}$, and by determining if the cell either partially intersects or totally belongs to the set $\mathcal{N}(\mathsf{x}, r)$. The cell $c^{j_0}_{k,\ell}$ is easily ascertained to satisfy this property by choosing the indices such that $(k, \ell) \in (\mathcal{K}(j_0), \mathcal{L}(j_0))$. If the cell needs to be subdivided into higher resolution cells, the inverse fast lifting wavelet transform is first performed on the current cell (local reconstruction) in order to recover the four approximation coefficients at level $j_0 + 1$ and the corresponding detail coefficients. We then adopt the raster scan method [23] (zigzag search: I→II→III→IV) to examine each cell inside the parent cell overlapping with $\mathcal{N}(\mathsf{x}, r)$. This procedure is recursively repeated until the maximum resolution level $J_{\max}$ is reached. Figure 5

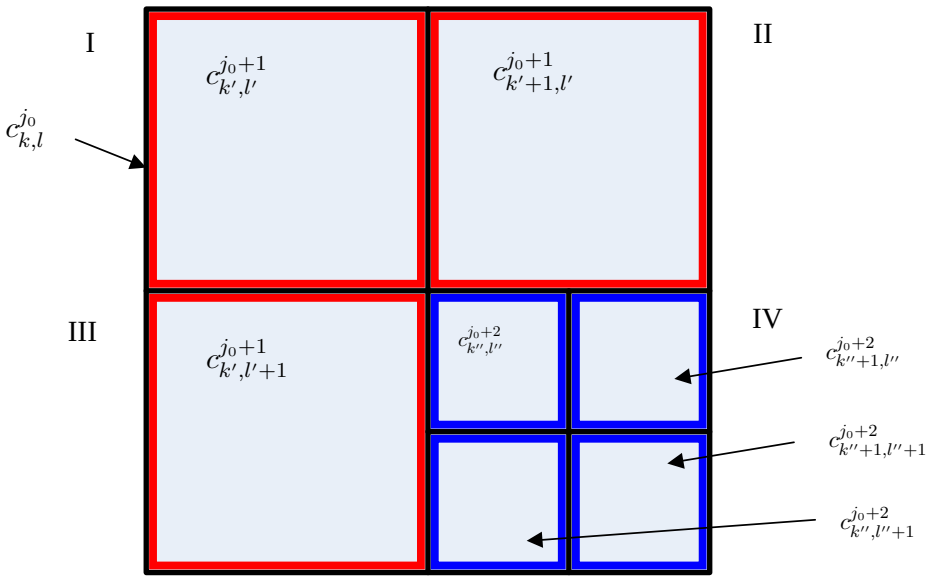


Fig. 4. Multiresolution cell subdivision across different levels.

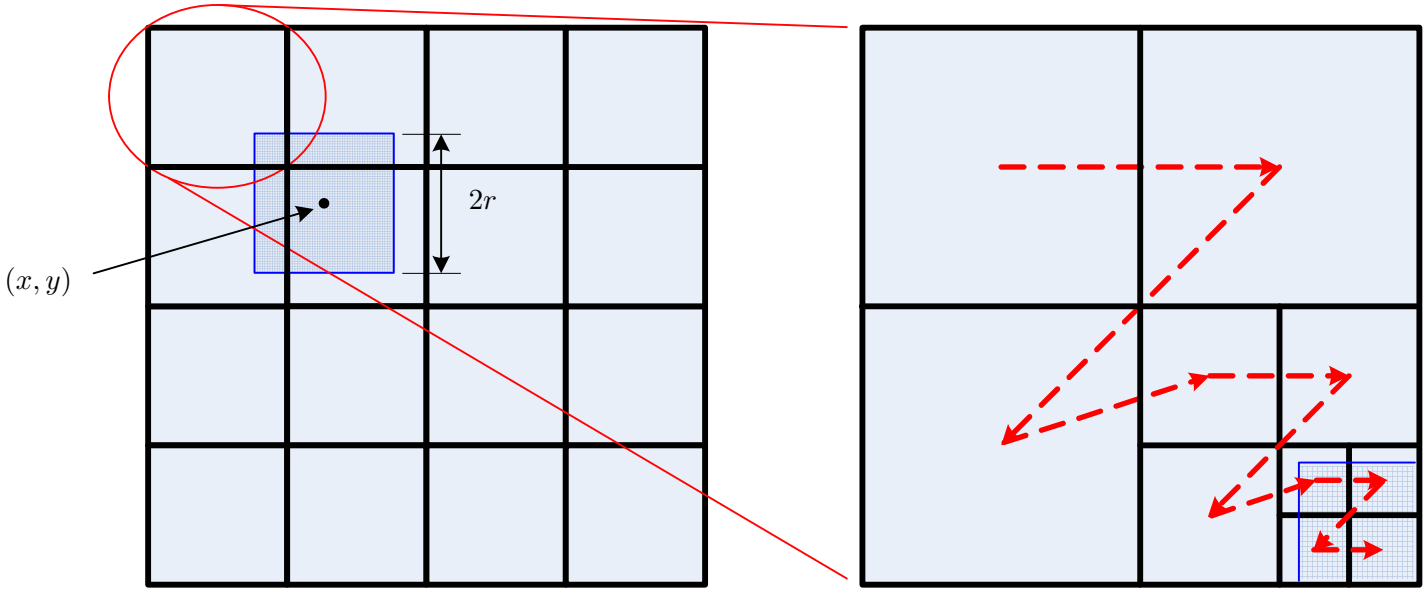


Fig. 5. Recursive raster scan method for identifying independent cells.

illustrates the recursive raster scan search. Once a cell is recognized as independent, we assign a node in the graph $\mathcal{G}$ with the node cost being the approximation coefficient representing the average risk measure over the cell. In addition, the detail coefficients associated with the current cell are all set to zero; this will provide the necessary connectivity information between the cells later on.

After a cell has been identified as an independent cell, we search the adjacent cells in order to establish the adjacency relationship with the current cell. Recall that two cells c_i and c_j are adjacent if

$$\partial c_i \cap \partial c_j \neq \varnothing, \quad i \neq j,$$

where ∂c_i denotes the boundary of the cell c_i. For our case of square cells, this implies that two cells are adjacent only along the following eight directions: Left, top, right, bottom, and the four diagonal directions. Following the recursive raster search for cell identification, the adjacency search requires establishing links between two cells that have been identified as independent cells. Recalling that the raster search progresses from left to right and from top to down (zigzag progress) as illustrated in Fig. 5, we confine the adjacency search to the following

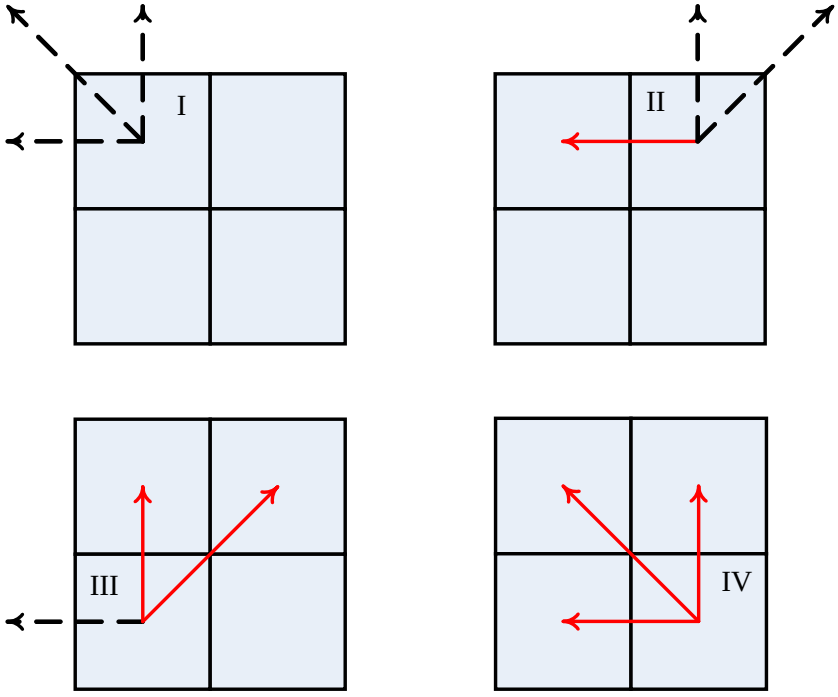


Fig. 6. Basic connectivity properties with respect to the location of the leaf cell.

directions: Left, top-left, top, and top-right from the current cell. By doing this, we render half of the links (out of eight connectivity) to be connected from the current cell, and the remaining links with the current cell will be connected as the recursive raster scan progresses to the next cells. In addition, because we deal with cells of different dimensions, it is required to devise a generic method to find the adjacency relationship between the cells.

Figure 6 illustrates the basic search direction of each leaf cell inside a parent cell. The dashed arrow points towards an external search region, that is, an adjacent cell could be found beyond the parent cell, whereas the solid arrow points towards an internal search region that belongs to the parent cell. In each search, we implicitly assume that the level of adjacent cells may vary from that of the parent cell to $J_{\max}$ (external connection), or from that of the current cell to $J_{\max}$ (internal connection).

A leaf cell inherits the search region from its parent cells, whose search direction turns out to be one of the solid arrows in Fig. 6. Figure 7 shows this inheritance property. In Fig. 7 the current cell is chosen to be $c_{\mathrm{I}}^{j_0+2}$. This cell is a leaf cell of the parent cell $c_{\mathrm{IV}}^{j_0+1}$, which further becomes a leaf cell of the top-most parent cell $c_{k,l}^{j_0}$. The cell $c_{\mathrm{IV}}^{j_0+1}$ is located on the fourth quadrant inside the top-most parent cell $c_{k,l}^{j_0}$ so that the search region for $c_{\mathrm{IV}}^{j_0+1}$ ends up with the internal searches at the level j_0+1, whose adjacency search property is inherited to the cell $c_{\mathrm{I}}^{j_0+2}$ for left, top-left, and top direction searches. Having ascertained the basic search directions, we refine the adjacent search looking for opposite cells which must be independent and adjacent to the current cell. Because the opposite cells of the current cell could have different dimensions, as depicted in Fig. 7, we establish links by examining the associated detail coefficients of the opposite cells. Along the left search direction of $c_{\mathrm{I}}^{j_0+2}$, as illustrated in Fig. 7, one finds that only one independent cell at level j_0+1 is linked to $c_{\mathrm{I}}^{j_0+2}$.

The adjacency search algorithm refines its search to the higher levels if the opposite cell is not an independent cell, that is, if it is comprised of finer cells. This refinement subsequently forces a search of cells of the finer dimension (level) which are neighboring to the current cell. Subsequently, the detail coefficients of the opposite cells are examined in order to find the next finer cell that is adjacent to the current cell. For the top-left search direction of $c_{\mathrm{I}}^{j_0+2}$, as illustrated in Fig. 8(a), the search process initially examines the cell $c_{\mathrm{I}}^{j_0+1}$ located at the top-left corner of the current cell through the corresponding detail coefficient. Provided that the detail coefficient associated with the cell $c_{\mathrm{I}}^{j_0+1}$ takes a non-zero value, the cell is assumed to be not an independent cell. Subsequently, the cell $c_{\mathrm{I}}^{j_0+1}$ is subdivided and the search process repeats at level j_0+2 when the opposite cell to the current cell becomes an independent adjacent cell. In Fig. 8(a), since there exists no other independent cells along the top-left

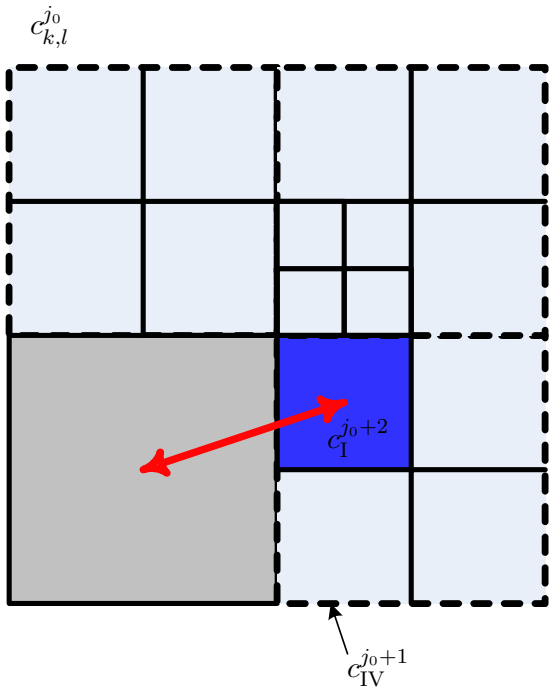


Fig. 7. Searching an adjacent cell along the left search direction.

direction except the shaded one, a bidirectional link is established between the current and the opposite cells.

Similarly, for the top search direction, two cells at level j_0+3 and one at level j_0+2 are found to be independent and adjacent to the current cell. The bidirectional links are accordingly connected from the current cell $c_{\mathrm{I}}^{j_0+2}$ to those adjacent cells. Figure 8(b) depicts this situation. Finally, Fig. 9 shows an example of the graph structure obtained from the multiresolution cell decomposition associated with the wavelet coefficients. Without loss of generality the nodes are located at the center of each cell. The solid lines show the connectivity relationship between the cells.

B. Cost assignment for $\mathcal{A}^*$ search

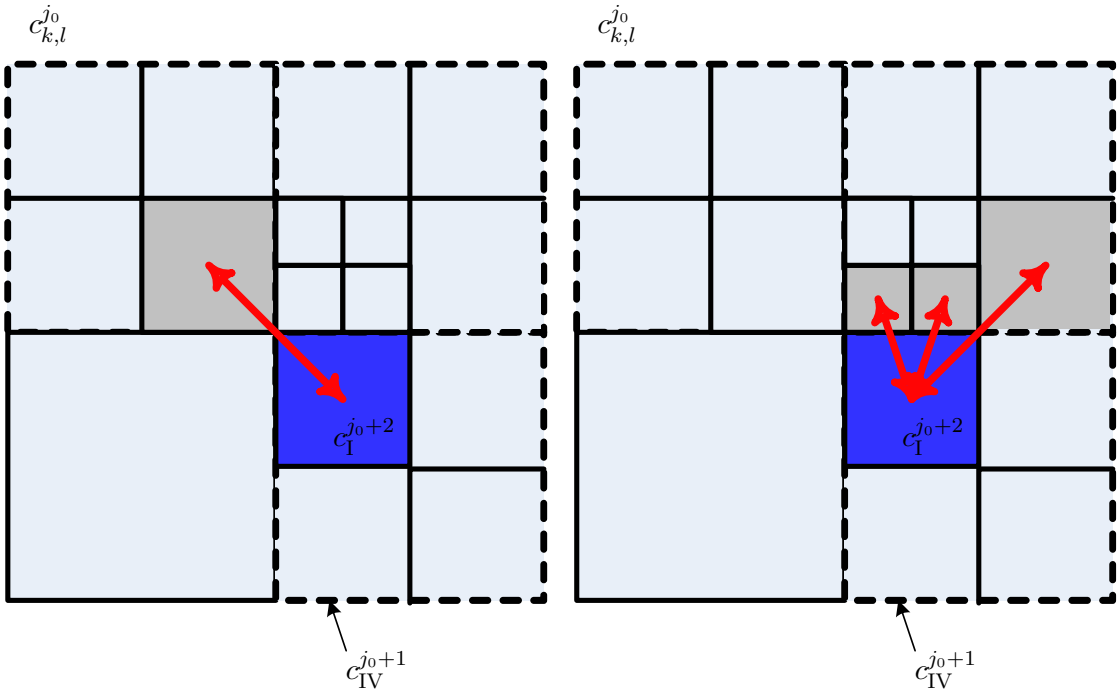
The $\mathcal{A}^*$ algorithm is a graph search algorithm that finds a path from an initial node to the goal node in the graph. The algorithm utilizes a *heuristic estimate* $h(v)$ that ranks each node v by a best cost estimate to reach the goal from the current node [24]. The algorithm visits the nodes in the order of the heuristic estimate, so the $\mathcal{A}^*$ algorithm is known as a best-first search algorithm. The key element of the $\mathcal{A}^*$ algorithm is that it expands each node from the priority queue that is ordered by (lower value has higher priority)

$$f(v) = g(v) + h(v), \tag{19}$$

where the cost $g(v)$ is the actual cost of the path up to v, i.e. the sum of the edge costs from the initial node, and $h(v)$ is the heuristic estimate at v. When a node u is expanded, the adjacent nodes to the current node are exploited. Let v be the adjacent node, then it follows that we evaluate the actual cost $g(v)$ to see if the transition from u to v results in lower cost than any other transitions to v. The algorithm then sets a back-pointer $\pi(v)$ by its preceding node u. This process iterates until the goal node is reached and no other nodes have a lower cost to the goal.

The $\mathcal{A}^*$ algorithm is complete in the sense that it is always guaranteed to find a solution if a solution exists. In addition, if the heuristic function $h(v)$ is admissible, that is, it uses an underestimate of the actual cost of reaching the goal, then $\mathcal{A}^*$ is optimal. Details about the implementation of the $\mathcal{A}^*$ algorithm can be found, for instance, in Ref. [25].

To the cell decomposition (18) we associate each node $v \in \mathcal{G}$ to a cell $c_{k,\ell}^j$. Moreover, since $\mathcal{G}$ is a topological graph, we may associate each node v with some point $\mathsf{x} \in c_{k,\ell}^j$. Without loss of generality, we choose the center of the cell. Let $\mathsf{cell}_{\mathcal{G}}(v)$ denote the center of the corresponding cell. If $\mathsf{x} \in c_{k,\ell}^j$ we will write $v = \mathsf{node}_{\mathcal{G}}(\mathsf{x})$.



(a) Search top-left (b) Search top

Fig. 8. Refined adjacency search algorithm.

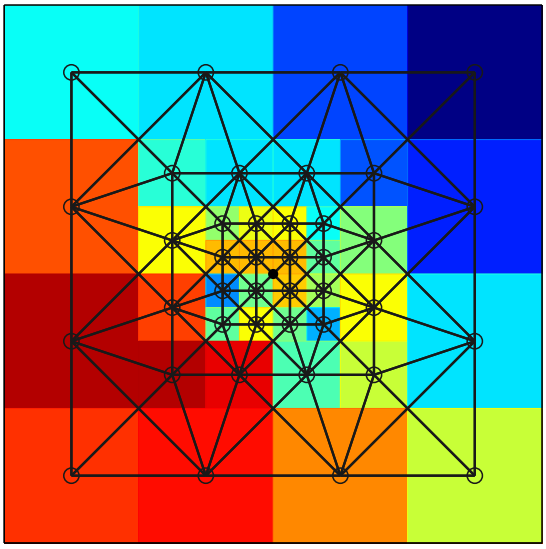


Fig. 9. Connectivity relationship constructed from the multiresolution cell decomposition over three levels.

To each directed edge (u, v) of $\mathcal{G}$ we assign an edge cost, given as

$$\mathcal{J}(u, v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \alpha \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, \tag{20}$$

where $\alpha \geq 0$ is a weight constant. The first term in (20) is proportional to the probability that the target node is close to an obstacle, while the second term penalizes the (Euclidean) distance between $\mathsf{cell}_{\mathcal{G}}(u)$ and $\mathsf{cell}_{\mathcal{G}}(v)$.

Suppose now that we are given a path of $q + 1$ consecutive, adjacent nodes in $\mathcal{G}$ as follows

$$\mathcal{P} = (v_0, v_1, \cdots, v_q). \tag{21}$$

We can then assign a traverse cost to each node in the path $\mathcal{P}$, induced by

$$g(v_i) = g(v_{i-1}) + \mathcal{J}(v_{i-1}, v_i), \qquad i = 1, \cdots, q. \tag{22}$$

The value of $g(v_k)$ represents the (accumulated) cost of the path from v_0 to v_k $(k \leq q)$, i.e. the weight of the edges followed up to v_k. We use the following heuristic estimate

$$h(v) = \|\mathsf{cell}_{\mathcal{G}}(v) - \mathsf{cell}_{\mathcal{G}}(v_f)\|_\infty, \tag{23}$$

where $v_f = \mathsf{node}_{\mathcal{G}}(\mathsf{x}_f)$.

The $\mathcal{A}^*$ algorithm then finds a path that minimizes the cost in (22) to the final node, or determines that such a path does not exist.

IV. Multiresolution Path Planning

A. Multiresolution path planning algorithm

The proposed multiresolution path planning algorithm proceeds as follows. Starting from $\mathsf{x}(t_0) = \mathsf{x}_0$ at time $t = t_0$, we construct using the approach of Section II a cell decomposition $\mathcal{C}_d(t_0)$ of $\mathcal{W}$. A topological graph, and the adjacency list of its nodes are obtained using the approach of Section III. Let the corresponding graph be $\mathcal{G}(t_0)$ and let $v_1^0 \in \mathcal{G}(t_0)$ and $v_f^0 \in \mathcal{G}(t_0)$ be the initial and the goal nodes such that $v_1^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_0)$ and $v_f^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_f)$, respectively. Using the $\mathcal{A}^*$ algorithm we compute a path $\mathcal{P}(t_0)$ in $\mathcal{G}(t_0)$ of free and mixed nodes from v_1^0 to v_f^0 assuming that such a path exists. Let $\mathcal{P}(t_0)$ be given by an ordered sequence of $l_0 + 1$ nodes as follows

$$\mathcal{P}(t_0) = (v_0^0,\ v_1^0, \cdots,\ v_{l_0-1}^0,\ v_{l_0}^0 = v_f^0). \tag{24}$$

It is assumed that v_1^0 is a free node owing to the high resolution representation of $\mathcal{W}$ close to x_0. The agent subsequently moves from v_0^0 to v_1^0. Let now t_1 be the time the agent is at the location $\mathsf{x}(t_1) = \mathsf{cell}_{\mathcal{G}(t_0)}(v_1^0)$ and let $\mathcal{C}_d(t_1)$ be the multiresolution cell decomposition of $\mathcal{W}$ around $\mathsf{x}(t_1)$ with a corresponding topological graph $\mathcal{G}(t_1)$. Applying again the $\mathcal{A}^*$ algorithm we compute a (perhaps new) path in $\mathcal{G}(t_1)$ from $v_0^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1))$ to $v_f^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}_f)$ if such a path exists. Let $\mathcal{P}(t_1)$ be given by the ordered sequence of $l_1 + 1$ nodes as follows

$$\mathcal{P}(t_0) = (v_0^1,\ v_1^1, \cdots,\ v_{l_1-1}^1,\ v_{l_1}^1 = v_f^1). \tag{25}$$

The agent subsequently moves to v_1^1 at location $\mathsf{x}(t_2) = \mathsf{cell}_{\mathcal{G}(t_1)}(v_1^1)$ at time t_2.

In general, assume the agent is at location $\mathsf{x}(t_i)$ at time t_i. We construct a multiresolution decomposition $\mathcal{C}_d(t_i)$ of $\mathcal{W}$ around $\mathsf{x}(t_i)$ with a corresponding graph $\mathcal{G}(t_i)$. The $\mathcal{A}^*$ algorithm yields a path $\mathcal{P}(t_i)$ in $\mathcal{G}(t_i)$ of length $l_i + 1$,

$$\mathcal{P}(t_i) = (v_0^i,\ v_1^i, \cdots,\ v_{l_i-1}^i,\ v_{l_i}^i = v_f^i), \tag{26}$$

where $v_0^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}(t_i))$ and $v_f^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}_f)$ if such a path exists. This iteration process terminates at some time t_f when $\|\mathsf{x}(t_f) - \mathsf{x}_f\|_\infty < 1/2^{J_{\max}}$. At the last step the agent moves from $\mathsf{x}(t_f)$ to x_f. A pseudo code implementation of the multiresolution path planning algorithm is given in Fig. 10. Note that the actual path followed by the agent is given by the sequence of nodes $\big\{\mathsf{node}_{\mathcal{G}(t_0)}\big(\mathsf{x}(t_0)\big), \mathsf{node}_{\mathcal{G}(t_1)}\big(\mathsf{x}(t_1)\big), \cdots, \mathsf{node}_{\mathcal{G}(t_f)}\big(\mathsf{x}(t_f)\big)\big\}$.

B. $\mathcal{D}^*$ lite path planning algorithm

The $\mathcal{D}^*$ algorithm has been originally proposed by Stentz [26], [27] for planning a path in unknown or partially known environments. Prior to $\mathcal{D}^*$, several replanning strategies have been proposed to solve dynamic planning problems for locally-directed wandering [28], local modification of initial path [29], and obstacle perimeter detouring [30]. Although these methods are complete, they are suboptimal and computationally inefficient. On the contrary, $\mathcal{D}^*$ produces an optimal path by adopting an efficient incremental search to reduce the time required to replan. In particular, $\mathcal{D}^*$ is more appropriate when dealing with an environment having a large number of states, reusing information from the previous search to find the solution at the next step. Koenig and Likhachev introduced Lifelong Planning $\mathcal{A}^*$ (LPA) [31] which employs heuristic estimate like $\mathcal{A}^*$, while reusing information from previous searches to find a solution much faster than solving each iteration from scratch. Furthermore, Koenig and Likhachev presented the $\mathcal{D}^*$-lite algorithm, derived from the LPA algorithm, which implements the same planning strategy as $\mathcal{D}^*$ but is algorithmically different. The $\mathcal{D}^*$-lite algorithm simplifies the maintenance of priority queues, thus it does not use complicated conditional statements, thus ending up with shorter codes than the original $\mathcal{D}^*$ algorithm. In the sequel, we employ the $\mathcal{D}^*$-lite algorithm to the path planning problem on a non-trivial environment. By comparing

```
BEGIN PATH PLANNING ALGORITHM
{
    i = 0;
    x(t_i) ← x_0;
    while ‖x(t_i) − x_f‖ ≥ 1/2^{J_max}
    {
        compute rm(x, i) for all x ∈ W;
        construct C_d(i) at level J_min;
        construct G(i) = (E(i), V(i));
        v_1^i ← node_{G(i)}(x(t_i));
        v_f^i ← node_{G(i)}(x_f);
        P(i) ← Astar(v_1^i, v_f^i, G(i));
        if P(i) = ∅
            report FAILURE; break;
        x(t_{i+1}) ← cell_{G(i)}(v_2^i);
        Move to x(t_{i+1});
        i ← i + 1;
    }
}
END PATH PLANNING ALGORITHM
```

Fig. 10. Pseudo-code implementation of proposed multiresolution path planning scheme.

the $\mathcal{D}^*$-lite algorithm with the multiresolution path planning algorithm, we discuss the benefits and shortfalls of using the multiresolution path planning algorithm over the $\mathcal{D}^*$-lite algorithm.

We apply the Haar wavelet transform up to resolution level $J \geq J_{\min}$ to obtain the wavelet decomposition of rm, given by

$$\mathsf{rm}(x,y) = \sum_{k,\ell=0}^{2^J-1} a_{J,k,\ell}\Phi_{J,k,\ell}(x,y) + \sum_{i=1}^{3}\sum_{j=J}^{N-1}\sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell}\Psi^i_{j,k,\ell}(x,y). \tag{27}$$

A uniform cell decomposition $\mathcal{C}_d^J$ at level J on $\mathcal{W}$ is induced from Eq. (27), and is comprised of cells $c^J_{k,\ell}$ of dimension $1/2^J \times 1/2^J$. We adopt the eight-connectivity relationship between the cells. The connectivity relationship is easily found by bookkeeping the location of each cell through the indices k and ℓ. It should be noted that the adjacency relationship will remain the same throughout the replanning, but the edge costs will change incrementally to incorporate the information from the previous step.

Suppose the agent is equipped only with a proximity sensor that senses the environment close to the current location with high accuracy. That is, the sensor provides information classifying the neighboring environment into a free region or obstacle region. Let $\mathcal{S}(i)$ be the known region up to $t = t_i$ using a sensor with the range r_J, defined by

$$\mathcal{S}(i) = \mathcal{S}(i-1) \cup \mathcal{N}(\mathsf{x}_i, r_J) \tag{28}$$

where x_i is the current location of the agent at $t = t_i$ and $\mathcal{N}(\mathsf{x}_i, r_J)$ represents the effective sensory region at that moment. In Eq. (28) it is assumed that the agent navigates an initially unknown environment while updating the map from the collected information. In order to take this process into consideration for replanning, we assign a conditional cost to each edge (u, v) which depends on the relative location of the edges to $\mathcal{S}$ as follows,

$$\mathcal{J}(u,v) = \begin{cases} \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, & \text{if } u,\ v \in \mathcal{S}, \\ \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2, & \text{if } u,\ v \in \mathcal{W} \setminus \mathcal{S}. \end{cases} \tag{29}$$

It follows that for the edges outside $\mathcal{S}$ we simply impose the traversal cost between nodes owing to the uniform size of cells. If this is the case, a general path planning algorithm such as Dijkstra's or $\mathcal{A}^*$ simply computes a shortest path from an initial node to the goal node which might pass through obstacles outside $\mathcal{S}$. Nevertheless, whenever

```
BEGIN D* LITE PATH REPLANNING ALGORITHM
{
    i = 0;
    compute rm(x) for all x ∈ W;
    construct C_d at level J_max;
    construct G = (E, V);
    v_start ← node_G(x_0);
    v_goal ← node_G(x_f)                                                    ;
    Initialize();
    ComputeShortestPath(v_start, v_goal, G);
    v_i ← v_start;
    v_last ← v_i;
    while (v_i ≠ v_goal)
    {
        i ← i + 1;
        v_i = argmin_{v'∈Adj(v_last)} (J(v_last, v') + g(v'));
        Move to v_i;
        Scan graph for changed edge costs;
        If any edge costs changed;
        {
            k_m ← k_m + h(v_last, v_i);
            v_last ← v_i;
            For all directed edges (u, v) ∈ E with changed edge costs
            {
                Update the edge cost J(u, v);
                UpdateVertex(v);
            }
            ComputeShortestPath(v_i, v_goal, G);
        }
    }
}
END D* LITE PATH REPLANNING ALGORITHM
```

Fig. 11. Pseudo-code implementation of $\mathcal{D}^*$ lite path planning scheme.

the map is updated using contingent information from the sensor, we accordingly update the corresponding edge costs by appending the obstacle cost to each edge as given in (29).

The main $\mathcal{D}^*$-lite path planning algorithm proceeds as follows. From the uniform cell decomposition and the corresponding graph, we solve for an initial path from $v_0 = v_{\text{start}}$ to v_{goal} assuming only the distance cost for edge weights. Let v_1 be the node next to v_0 in the path. The agent subsequently moves from v_0 to v_1. At time $t = t_1$ when the agent is located at v_1, the algorithm continues to scan the graph for changed edge costs. If any edge costs have changed, then the algorithm updates the corresponding edge weights. Finally, a new path is computed from v_1 to v_{goal}, while incorporating the updated edge weights. It should be noted that if no edge costs have changed the agent moves to the successive node v' in the previous path that has the minimum cost $\mathcal{J}(v_{\text{last}}, v') + g(v')$.

Similar to $\mathcal{A}^*$, the $\mathcal{D}^*$-lite algorithm also incorporates a heuristic estimate to choose the nodes from a priority queue. However, as the agent detects changes in the edge costs, the priority queue is reordered to render itself consistent. This might be an expensive task, so instead of reordering the priority queue every time, Koenig and Likhachev utilizes a *dynamic heuristic constant* k_m [32] to keep the priority queue unaltered regardless of the change of the edge costs. The iteration terminates at some time t_l when the goal node is reached. A pseudo code implementation of the $\mathcal{D}^*$ lite incremental path planning algorithm is given in Fig. 10. Note that the actual path followed by the agent is given by the sequence of nodes $\{v_0 = v_{\text{start}}, v_1, \cdots, v_{l-1}, v_l = v_{\text{goal}}\}$.

V. Hardware in-the-loop Simulation Results

A. Hardware overview

A UAV platform based on the airframe of an off-the-shelf R/C model airplane has been developed to implement the multiresolution, wavelet-based path planning algorithm described above. The development of the hardware and software was done completely in-house to have a full access of the entire system. The on-board autopilot is equipped with a micro-controller, sensors and actuators, along with the communication devices that allow full functionality for autonomous control. The micro-controller (Rabbit RCM-3400 running at 30 MHz with 512 KB RAM) provides data acquisition, processing, and communication with the ground station. It also runs the low-level control loops for basic stabilization and way-point navigation. The on-board sensors include angular rate sensors for three axes, accelerometers along all three axes, a three-axis magnetic compass, a GPS sensor, and absolute and differential pressure sensors. Figure 12 shows the UAV platform with the on-board autopilot.

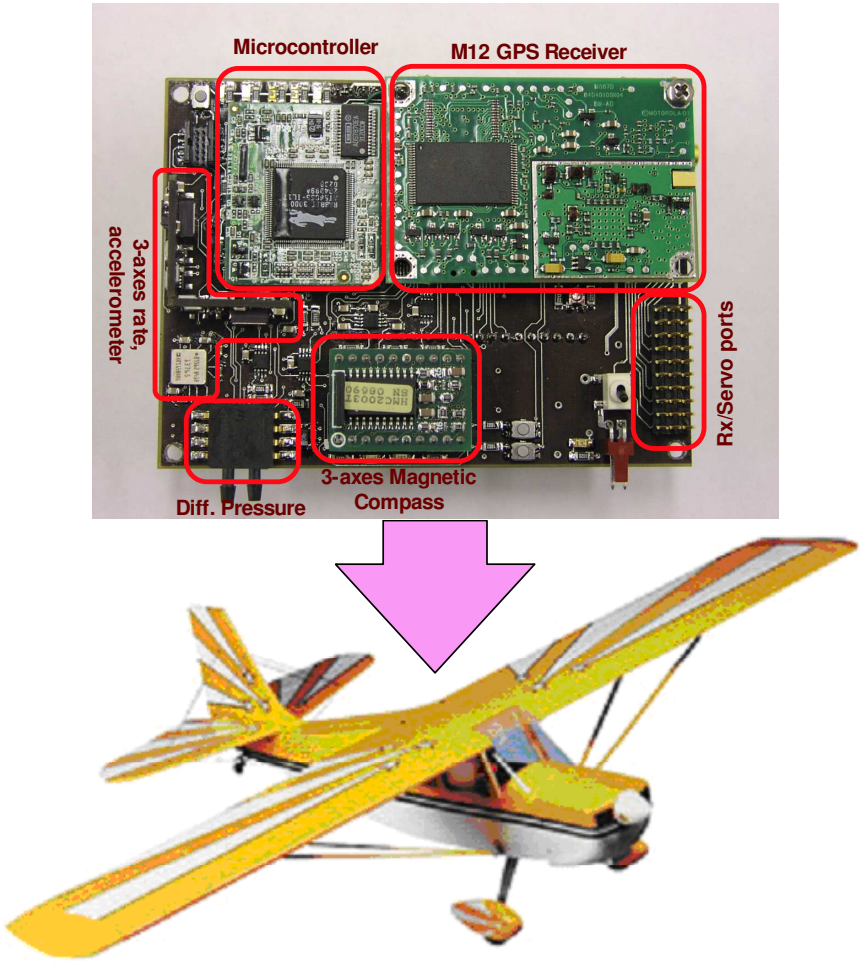


Fig. 12. A small fixed-wing UAV equipped with an autopilot for hierarchical path planning control.

A realistic hardware-in-the-loop simulation (HILS) environment has also been developed to validate the UAV autopilot hardware and software development utilizing Matlab® and Simulink®. A full 6-DOF nonlinear aircraft model is used in conjunction with a linear approximation of the aerodynamic forces and moments, along with Earth gravitational (WGS-84) and magnetic field models. Detailed models for the sensors and actuators have also been incorporated. Four independent computer systems are used in the hardware-in-the-loop simulation (HILS) as illustrated in Fig. 13. A 6-DOF simulator, the flight visualization computer, the autopilot micro-controller, and the ground station computer console are involved in the simulation. Further details about the UAV platform, autopilot and HILS set-up can be found in [33], [34] and [35].

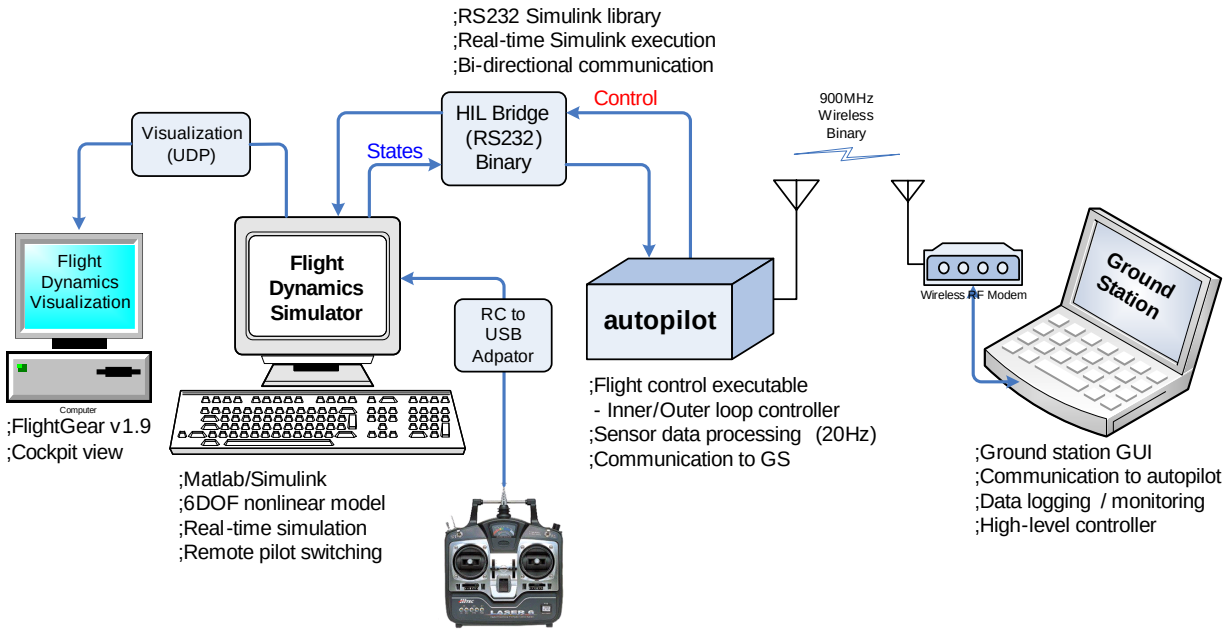


Fig. 13. High fidelity hardware-in-the-loop simulation (HILS) environment that enables rapid testing of the proposed path planning algorithm.

B. Simulation results for the proposed algorithm

In this section we present simulation results of the proposed algorithm for a non-trivial scenario. The environment $\mathcal{W}$ is an actual topographic (elevation) map of a certain US state with fractal-like characteristics, shown in Fig. 14. The environment is assumed to be square of dimension 128×128 units. Hence the finest possible resolution is $N = 7$. Taking into account the available memory of the micro-controller, we choose the fine level as $J_{\max} = 6$ and the coarse level as $J_{\min} = 3$. This makes the total number of nodes in the graph not exceed the maximum count of 256 that corresponds to the maximum allowable variable size of the micro-controller. The ranges from the current location at distinct levels of resolution are selected as follows,

$$(r_6, r_5, r_4) = (8, 15, 30) \quad \text{in units},$$

which dictates that the higher resolution representations around the current location of the agent are used inside an area of 60×60 unit cells at level $J = 4$ down to an area of 16×16 unit cells at the finest level $J_{\max} = 6$.

The objective of the UAV is to follow a path from the initial position to the final position while circumventing the obstacles over a certain elevation threshold. Since the on-line path planning problem at the finest resolution is computationally prohibitive, the proposed algorithm accommodates the need for the on-line implementation on the micro-controller by limiting the amount of the information to process, thus computing an immediate path with high accuracy within the allowable time scale of the micro-controller.

The results from the multiresolution path planning algorithm are shown in Fig. 14. Specifically, Fig. 14 shows the evolution of the path at different time steps as the agent moves to the final destination. Figure 14(a) shows the agent's position at time step $t = t_5$ along with the best proposed path to the final destination by a dashed-dot line at that time. Similarly, Fig. 14(b) shows the agent's position at time step $t = t_{21}$. As seen in Fig. 14(c), the actual path followed by the agent differs from the one predicted in either Figs. 14(a) or 14(b). This is due to the fact that at time t_5 and t_{21} the agent does not have complete information for upcoming positions up to confident level. In particular, as the agent gets closer to the obstacle as shown in Fig. 14(b), it recognizes the presence of obstacles and redirects the path to avoid any obstacles. The agent reaches the final destination $\times_f$ in a collision free manner, as seen in Fig. 14(c).

C. Simulation results for the $\mathcal{D}^*$-lite algorithm

In this section we present simulation results of the $\mathcal{D}^*$-lite path planning algorithm for the same environment used in the previous section. It is assumed that the agent navigates over the unknown environment, while updating

TABLE I
COMPUTATIONAL COST OF THE PROPOSED ALGORITHM BY THE ON-BOARD AUTOPILOT.

Multiresolution cell decomposition using FLWT	452 [msec]
Construct the connectivity relationship and $\mathcal{G}$	292 [msec]
Compute a path using $\mathcal{A}^*$ employing the binary heap	202 [msec]
Average number of nodes of each $\mathcal{G}$	$\sim$200

the map with the information gathered from a proximity sensor. We adopt a uniform cell decomposition of cell size the $J_{\max} = 6$ which is the same as the finest level of the previous section. The range of the proximity sensor is chosen to be $r_6 = 7$, thus resulting in the high resolution window by a 7 by 7 square grids.

The results from the $\mathcal{D}^*$-lite path planning algorithm are shown in Fig. 15. Specifically, Fig. 15 shows the evolution of the path at different time steps as the agent moves to the final destination. At each step, the best proposed path is drawn by a dashed-dot line and the actual path followed by the agent is drawn by a solid line. As seen in Fig. 15(c), the actual path differs significantly from the one predicted in either Figs. 15(a) or 15(b). This is attributed to the fact that the environment is unknown a priori, and the path is computed using the distance cost outside the high resolution area. Hence, as shown in Fig. 15(a), the agent is unable to anticipate the existence of the obstacles outside the high resolution area. Nonetheless, as the agent gets closer to the obstacles, the $\mathcal{D}^*$-lite algorithm effectively replans the entire path circumventing the obstacles, reaching the final destination.

VI. COMPARISON

The proposed multiresolution path planning algorithm was written in C code and implemented on the on-board autopilot equipped with a Rabbit RCM-3400 micro-controller. Because the micro-controller has limited computational resources (10,000 instructions per second, and 512 KB RAM for handling variables), the code has been written giving special attention not only to the accuracy of the output, but also to the computational speed during implementation. Specifically, most of the computations for the proposed algorithm is done using integer arithmetic. Given a risk measure rm of integer samples, the integer fast lifting wavelet transform provides the approximation and detail coefficients that are used to construct the adjacency relationship between cells. The $\mathcal{A}^*$ algorithm is then called to find the shortest path in this graph.

Table I shows the computational cost of the proposed path planning algorithm using the on-board autopilot. One step of the path planning iteration takes 946 [msec] for execution. Ascertaining the execution time of the proposed algorithm, we actually choose to implement the proposed path planning algorithm on-line in every three second. Hence, the autopilot manages not only to execute the basic tasks such as data acquisition and processing, inner loops control, and etc., but also to plan a path in a seamless manner.

We compared the computational costs between the proposed multiresolution path planning algorithm and the $\mathcal{D}^*$-lite algorithm, using different simulation results for several cases. The simulations were carried out on an IBM-PC (Pentium M 2.0 GHz, 1 GB RAM), based on codes written in C for implementing both path planning algorithms. The proposed path planning algorithm accomplishes the path planning objective of reaching the goal in less number of iterations, as shown in Tab. II, than the $\mathcal{D}^*$-lite algorithm. This is due to the fact that the proposed algorithm effectively manages the information at coarse resolutions so as to compute a preferred path. The $\mathcal{D}^*$-lite algorithm, however, relies on the information at finer resolution that is unveiled up to the current time, thus requiring the agent to explore the environment and to replan the path along the movement of the agent. In the worst case, the total number of iterations by the $\mathcal{D}^*$-lite algorithm increases significantly (e.g. Scenario IV) because of the existence of unknown obstacles.

The total computation time of the proposed algorithm is obtained by adding the computation times throughout each iteration. For the $\mathcal{D}^*$-lite algorithm, the total computation time consists of the time for initializing and updating, which is shown to be smaller than that of the proposed algorithm. The $\mathcal{D}^*$-lite algorithm is computationally efficient in the sense that it reuses information from the previous step. Most of the computations in the proposed algorithm are devoted to the construction of the adjacency list at each planning step, as shown in Tab. II. The performance of the proposed algorithm can thus be improved by using, say, four-connectivity instead of eight-connectivity during the adjacency search algorithm. This will possibly halve the computation time with little performance degradation. On the other hand, the proposed algorithm requires little memory as shown in Tab. II compared to $\mathcal{D}^*$-lite. For

TABLE II
THE COMPUTATIONAL COST COMPARISON BETWEEN THE MULTIRESOLUTION PATH PLANNING V/S THE $\mathcal{D}^*$ LITE.

Items	Scenario I		Scenario II		Scenario III		Scenario IV		Scenario V	
	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet	$\mathcal{D}^*$-lite	Wavelet
# iteration	35	31	93	50	61	40	123	52	44	43
# nodes in $\mathcal{G}$	4096	201	4096	194	4096	192	4096	185	4096	194
Data processing [msec]	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03	2.03
Adjacency search [msec]	-	0.977	-	0.987	-	0.969	-	0.94	-	0.958
$\mathcal{A}^*$ search [msec]	-	0.1	-	0.125	-	0.094	-	0.138	-	0.066
Init. $\mathcal{D}^*$-lite search [msec]	1.87	-	2.03	-	2.03	-	1.87	-	2.03	-
$\mathcal{D}^*$-lite update [msec]	4.1	-	23.8	-	11.3	-	43.9	-	12.7	-
Total Comp. time [msec]	5.97	33.387	25.83	55.6	13.33	42.53	45.77	56.056	14.73	44.032
Computational cost (%)	17.8	100	46.46	100	31.35	100	81.65	100	33.45	100
Memory cost (%)	2037.8	100	2111.3	100	2133.3	100	2214.1	100	2111.3	100

on-line, on-board path planning, the proposed algorithm has the advantages of scalability according to the available on-board computational resources.

VII. CONCLUSIONS

Autonomous path planning for small UAVs imposes severe restrictions on control algorithm development, stemming from the limitations imposed by the on-board hardware and the requirement for on-line implementation. In this work we have proposed a method to overcome this problem by using a new hierarchical, multiresolution path planning scheme. The algorithm computes at each step a multiresolution representation of the environment using the wavelet transform. The idea is to employ high resolution close to the agent where is needed most, and a coarse resolution at large distances from the current location of the agent. As an added benefit, the connectivity relationship of the resulting cell decomposition can be computed directly from the nonzero detail coefficients of the wavelet transform. The algorithm is scalable and can be tailored to the available computational resources of the agent.

REFERENCES

[1] Brooks, R. A. and Lozano-Peres, T., "A Subdivision Algorithm on Configuration Space for Findpath with Rotation," *IEEE Transactions on Systems Man Cybernet*, Vol. 15, 1985, pp. 224–233.

[2] Zhu, D. and Latombe, J.-C., "New Heuristic Algorithms for Efficient Hierarchical Path Planning," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, Feb. 1991, pp. 9–20.

[3] Godbole, D., Samad, T., and Gopal, V., "Active Multi-Model Control for Dynamic Maneuver Optimization of Unmanned Air Vehicles," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, 2000, pp. 1257–1262.

[4] Pai, D. K. and Reissell, L.-M., "Multiresolution Rough Terrain Motion Planning," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 1, 1998, pp. 19–33.

[5] Behnke, S., "Local Multiresolution Path Planning," *RoboCup 2003: Robot Soccer World Cup VII*, Vol. 3020 of *Lecture Notes in Computer Science*, Springer, Berlin, 2004, pp. 332–343.

[6] Noborio, H., Naniwa, T., and Arimoto, S., "A Quadtree-Based Path-Planning Algorithm for a Mobile Robot," *Journal of Robotic Systems*, Vol. 7, No. 4, 1990, pp. 555–574.

[7] Kambhampati, S. and Davis, L. S., "Multiresolution Path Planning for Mobile Robots," *IEEE Journal of Robotics and Automation*, Vol. 2, No. 3, Sept. 1986, pp. 135–145.

[8] Chen, D. Z., Szczerba, R. J., and Uhran, J. J., "A Framed-Quadtree Approach for Determining Euclidean Shortest Paths in a 2-D Environment," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 5, 1997, pp. 668–681.

[9] Vörös, J., "Low-cost Implementation of Distance Maps for Path Planning using Matrix Quadtrees and Octrees," *Robotics and Computer Integrated Manufacturing*, Vol. 17, 2001, pp. 447–459.

[10] Tsiotras, P. and Bakolas, E., "A Hierarchical On-Line Path Planning Scheme using Wavelets," *Proceedings of the European Control Conference*, Kos, Greece, July 2007.

[11] Uytterhoeven, G., Roose, D., and Bultheel, A., "Wavelet transforms using lifting scheme," Technical report ita-wavelets report wp 1.1, Katholieke Universiteit Leuven, Belgium, April 1997.

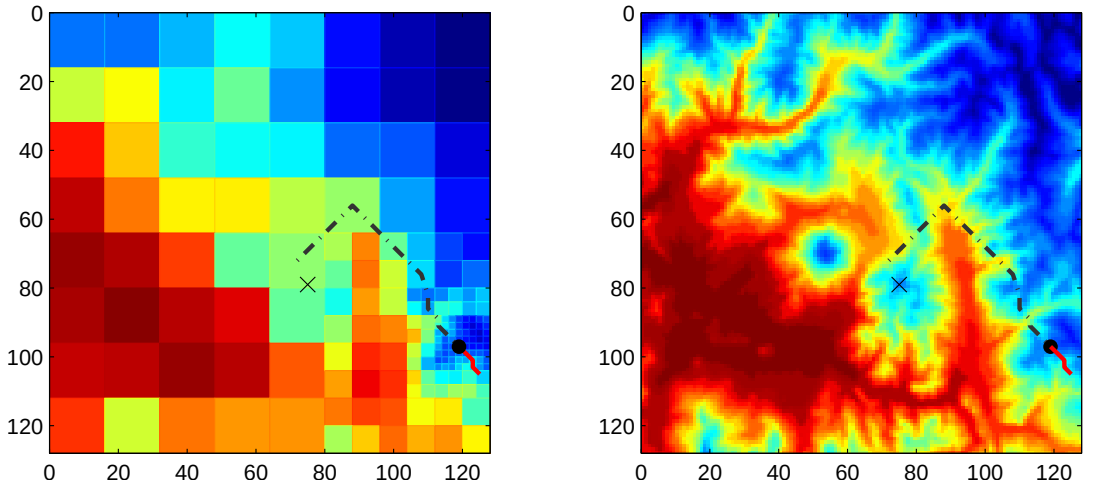
[12] Lozano-Perez, T. and Wesley, M. A., "Automatic Planning for Planning Collision-Free Paths Among Polyhedral Obstacles," *IEEE Transactions on System, Man, Cybernet*, Vol. 11, 1981, pp. 681–698.

[13] Burrus, C. S., Gopinath, R. A., and Guo, H., *Introduction to Wavelets and Wavelet Transforms*, Prentice Hall, Upper Saddle River, New Jersey, 1998.

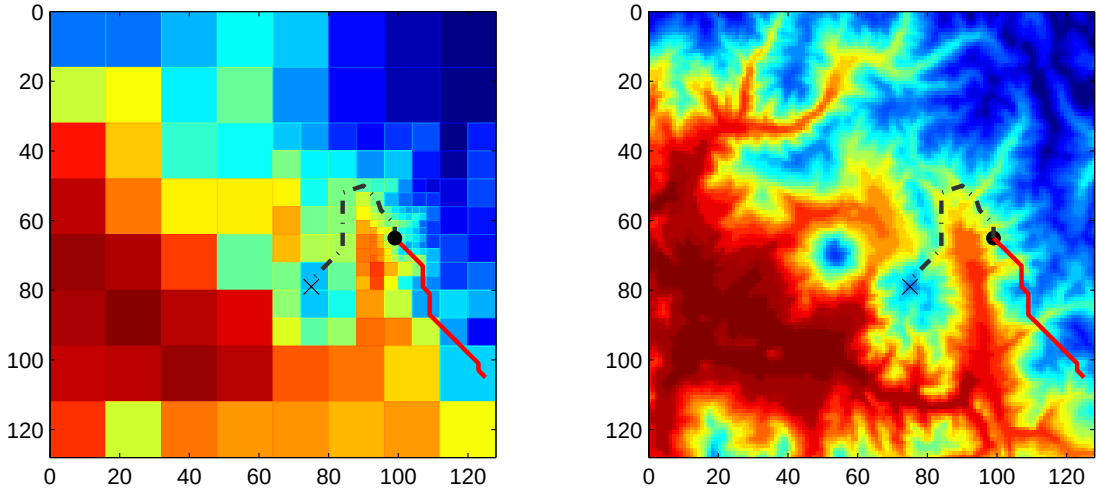
[14] Mallat, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 7, July 1989, pp. 674–693.

[15] Walnut, D. F., *An Introduction to Wavelet Analysis*, Birkhäuser, Boston, 2002.

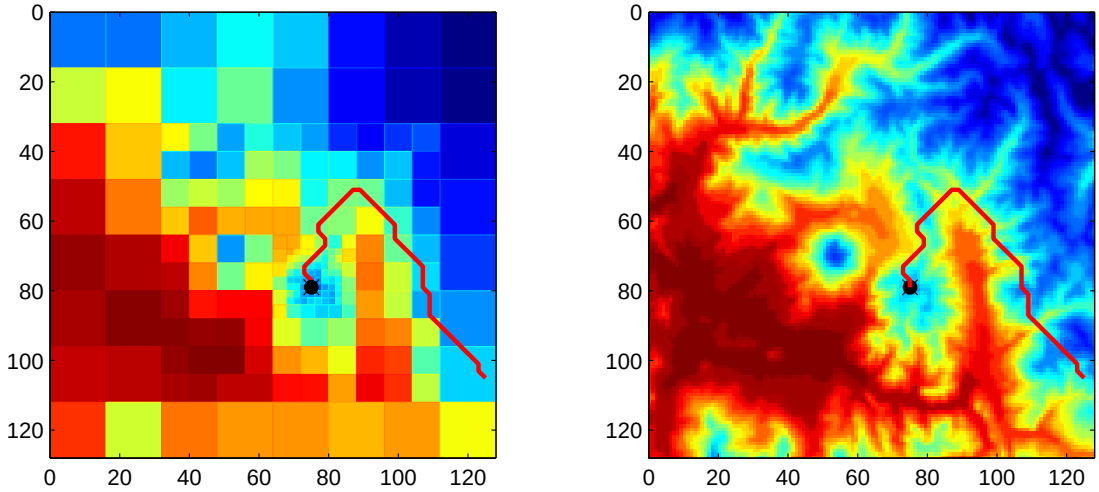
[16] Donoho, D. L., "Smooth Wavelet Decompositions with Blocky Coefficient Kernels," *Recent Advances in Wavelet Analysis*, Academic Press, 1993, pp. 1–43.
[17] Sweldens, W. and Schröder, P., "Building Your Own Wavelets at Home," *Wavelets in Computer Graphics*, ACM SIGGRAPH Course notes, 1996, pp. 15–87.
[18] Sweldens, W., "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," *Wavelet Applications in Signal and Image Processing III*, 1995, pp. 68–79.
[19] Sweldens, W., "The Lifting Scheme: A Construction of Second Generation Wavelets," *SIAM Journal on Mathematical Analysis*, Vol. 29, No. 2, 1997, pp. 511–546.
[20] Daubechies, I. and Sweldens, W., "Factoring Wavelet Transform into Lifting Steps," *Journal of Fourier Analysis and Applications*, Vol. 4, No. 3, 1998, pp. 247–269.
[21] Calderbank, A. R., Daubechies, I., Sweldens, W., and Yeo, B.-L., "Wavelet Transforms That Map Integers to Integers," *Applied and Computational Harmonic Analysis*, Vol. 5, No. 3, 1998, pp. 332–369.
[22] Heer, V. K. and Reinfelder, H.-E., "A Comparison of Reversible Methods for Data Compression," *Medical Imaging IV: Image Processing*, Vol. Proc. SPIE Vol. 1233, 1990, pp. 354–365.
[23] Mertz, P. and Gray, F., "A Theory of Scanning and Its Relation to the Characteristics of the Transmitted Signal in Telephotography and Television," *Bell System Technical Journal*, Vol. 13, 1934, pp. 464–515.
[24] Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, 1968, pp. 100–107.
[25] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*, The MIT Press, 2nd ed., Sept. 2001.
[26] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments," *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 4, May 1994, pp. 3310–3317.
[27] Stentz, A., "Map-based Strategies for Robot Navigation in Unknown Environments," *Proceedings AAAI 1996 Spring Symposium on Planning with Incomplete Information for Robot Problems*, Menlo Park, CA, 1996, pp. 110–116.
[28] Pirzadeh, A. and Snyder, W., "A Unified Solution to Coverage and Search in Explored and Unexplored Terrains using Indirect Control," *IEEE International Conference on Robotics and Automation*, Vol. 3, May 1990, pp. 2113–2119.
[29] Korf, R. E., "Real-Time Heuristic Search," *Artificial Intelligence*, Vol. 42, 1990, pp. 189–211.
[30] Lumelsky, V.; Stepanov, A., "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment," *IEEE Transactions on Automatic Control*, Vol. 31, No. 11, Nov. 1986, pp. 1058–1063.
[31] Koenig, S. and Likhachev, M., "Incremental $\mathcal{A}^*$," *Advances in Neural Information Processing Systems*, 2002, pp. 1539–1546.
[32] Koenig, S. and Likhachev, M., "D* Lite," *Proceedings of the National Conference of Artificial Intelligence*, 2002, pp. 476–483.
[33] Jung, D., Levy, E. J., Zhou, D., Fink, R., Moshe, J., Earl, A., and Tsiotras, P., "Design and Development of a Low-Cost Test-Bed for Undergraduate Education in UAVs," *Proceedings of the* 44^{th} *IEEE Conference on Decision and Control*, Seville, Spain, Dec. 2005, pp. 2739–2744.
[34] Jung, D. and Tsiotras, P., "Inertial Attitude and Position Reference System Development for a Small UAV," *AIAA Infotech at Aerospace*, Rohnert Park, CA, May 2007, AIAA Paper 07-2768.
[35] Jung, D. and Tsiotras, P., "Modelling and Hardware-in-the-loop Simulation for a Small Unmanned Aerial Vehicle," *AIAA Infotech at Aerospace*, Rohnert Park, CA, May 2007, AIAA Paper 07-2763.



(a) $t = t_5$

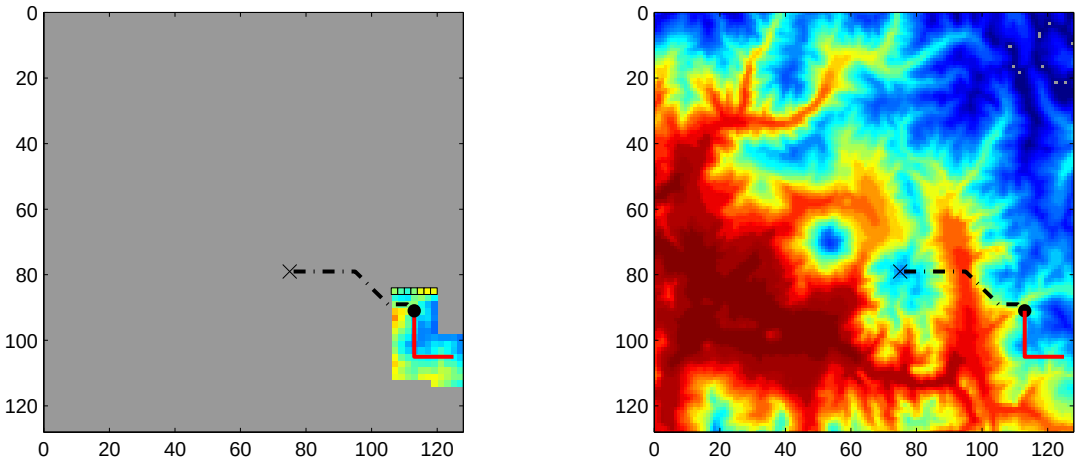


(b) $t = t_{21}$

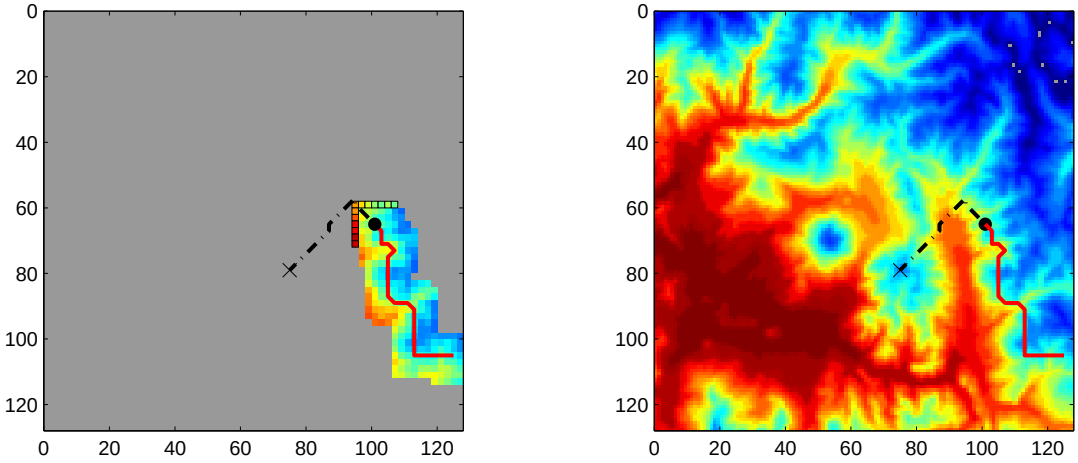


(c) $t = t_f$

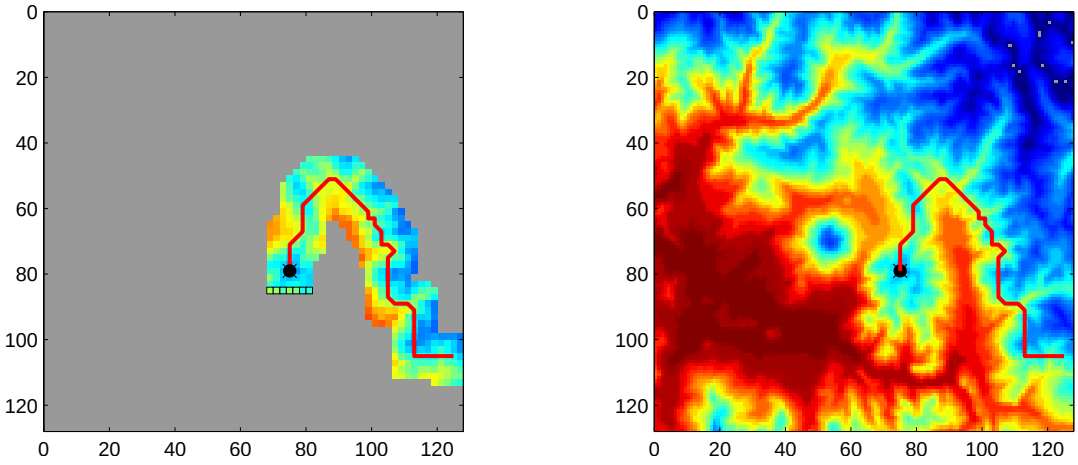
Fig. 14. Path evolution and replanning. Figures on the left show the currently tentative optimal path obtained from the $\mathcal{A}^*$ algorithm, based on the available multiresolution approximation of the environment at different time steps. Figures on the right show the actual path followed by the agent.



(a) $t = t_{14}$



(b) $t = t_{30}$



(c) $t = t_{53}$

Fig. 15. Path evolution and replanning using $\mathcal{D}^*$-lite algorithm. Figures on the left show the currently tentative optimal path obtained from the $\mathcal{D}^*$-lite algorithm, based on the distance cost outside the high resolution area. Figures on the right show the actual path followed by the agent.