Multiresolution Hierarchical Path-Planning for Small UAVs

Panagiotis Tsiotras
School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA 30332-0150, USA
Email: tsiotras@gatech.edu

***Abstract*— In this paper we review some recent results on a new multiresolution hierarchical path planning algorithm for mobile agents with limited on-board computational resources. The proposed approach assumes that the agent (e.g., UAV) has detailed knowledge of the environment and the obstacles only in its vicinity. Far away obstacles are only partially known. The algorithm uses the Fast Lifting Wavelet Transform (FLWT) to construct a graph representation of the environment, whose dimension is commensurate to the on-board computational resources of the agent. The adjacency list of this graph can be efficiently constructed directly from the approximation and detail wavelet coefficients without the need to resort to a separate quadtree decomposition, thus further speeding up the whole process. The optimal path is found by implementing a standard graph search algorithm such as Dijkstra'a algorithm or A*.**

Keywords: Mobile agent, wavelet decomposition, path planning, collision avoidance, adjacency matrix, UAVs

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 (CPU and memory) to implement some of the sophisticated path-planning algorithms proposed in the literature (Fig. 1). In most applications so far this problem is bypassed by providing navigation way points that have been computed off-line on the ground, or on-line by a more capable supervising/leader agent.

In a typical mission the UAV may be presented with a large amount of information collected using a plethora of diverse sensors (e.g., cameras, radars, laser scanners, satellite imagery) having different ranges and resolutions. A computationally efficient path planning algorithm should therefore be able to blend together the information provided by all these sensors, and focus its computational resources on 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 able to combine short-term tactics (reaction to unforeseen threats) with long-term strategy (planning towards the ultimate goal).

Several multi-resolution or hierarchical algorithms have been proposed in the literature to alleviate the computational burden associated with path planning over a complex, unstructured, and partially known environment [1], [2], [3], [4], [5], [6], [7]. The majority of those algorithms use some form of quadtree decomposition of the environment. 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.

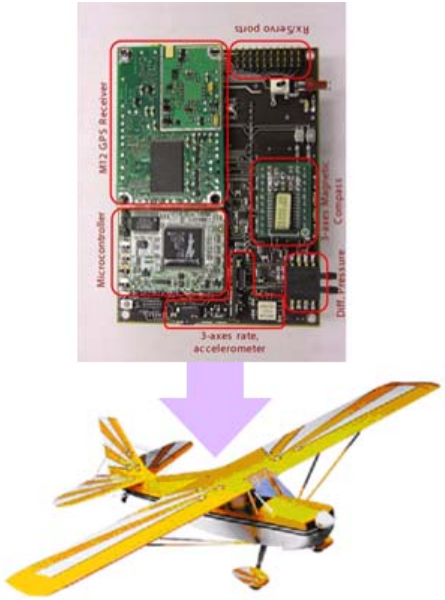


Fig. 1. Autonomous path generation and tracking for a small-scale UAV is limited by the available on-board computational resources. A practical path-planning algorithm for such small UAVs has to incorporate the limitations imposed by the hardware early on in the design process.

Recently, we proposed a computationally 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 our approach hinges on using district levels of fidelity (resolution) of $\mathcal{W}$ at different distances from the agent's current position. The motivation for this approach is simple: first, the agent's immediate reaction to an obstacle or a threat is needed only at the vicinity of its current position. Far away obstacles or threats do not (or should not) have a large effect on the vehicle's *immediate* motion. Therefore, it is not prudent from a computational point of view to find a solution with great accuracy over large ranges or over a very long time horizon. The most accurate and reliable information of the environment is required only at the vicinity of the vehicle.

We use the wavelet transform to perform the required multi-resolution decompositions of the environment. The

wavelet transform provides a very fast decomposition[1] of a function at different levels of resolution. In addition, the use of wavelet transform has the added benefit of allowing the construction of the associated cell connectivity relationship directly from the wavelet coefficients, thus eliminating the need for quadtrees (see Section III-A).

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 path may contain mixed nodes at all resolution levels except at the finer resolution level, where it is assumed that nodes can be confidently resolved as either free or occupied. Hierarchical path planning is known to be more flexible than methods that search only through free nodes [9].

Since the range and resolution levels can be chosen by the user, the proposed algorithm results in search graphs of known a priori complexity. The algorithm is hence scalable and can be tailored to the available computational resources of the agent.

In the sequel $\mathcal{W} \subset \mathbb{R}^2$ denotes the working environment that includes the free space and the obstacles, $\mathcal{F} = \mathcal{W} \backslash \mathcal{O}$ is the obstacle-free configuration space that contains all the feasible states of the agent, and $\mathcal{O} \subset \mathcal{W}$ is the obstacle space.

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

A. The 2D wavelet transform

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

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

where $\phi_{j,k}(x) = 2^{j/2} \phi(2^j x - k)$ and $\psi_{j,k} = 2^{j/2} \psi(2^j x - k)$. In the ideal case both $\phi(x)$ (scaling function) and $\psi(x)$ (mother wavelet) have compact support, or they decay very fast outside a small interval so they can capture localized features of f. The first summation in (1) gives a low resolution or coarse approximation of f. The second term in (1) gives the difference (details) between the original function and its low resolution approximation. For example, when analyzing a signal at the coarsest level (low resolution) only the general, most salient features of the signal will be revealed. The index j denotes the resolution level. For each increasing index j, a higher, or finer resolution term is added, which adds more and more details. The expansion (1) thus reveals the properties f at different levels of resolution [10], [11], [12].

This idea can be readily extended to the two-dimensional case by introducing the following families of functions

$$\Phi_{j,k,\ell}(x,y) = \phi_{j,k}(x)\phi_{j,\ell}(y) \qquad (2)$$
$$\Psi^1_{j,k,\ell}(x,y) = \phi_{j,k}(x)\psi_{j,\ell}(y) \qquad (3)$$
$$\Psi^2_{j,k,\ell}(x,y) = \psi_{j,k}(x)\phi_{j,\ell}(y) \qquad (4)$$
$$\Psi^3_{j,k,\ell}(x,y) = \psi_{j,k}(x)\psi_{j,\ell}(y) \qquad (5)$$

[1]The computational complexity of the wavelet transform is of order $O(n)$ where n is the input data [8]. This is better even than the Fast Fourier Transform which has complexity of order $O(n \log_2 n)$.

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

$$\begin{aligned} f(x,y) = &\sum_{k,\ell \in \mathbb{Z}} a_{J,k,\ell} \Phi_{J,k,\ell}(x,y) \\ &+ \sum_{i=1}^{3} \sum_{j \geq J} \sum_{k,\ell \in \mathbb{Z}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y), \end{aligned} \qquad (6)$$

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

$$a_{j,k,\ell} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, \Phi_{j,k,\ell}(x,y)\, \mathrm{d}x\, \mathrm{d}y, \qquad (7)$$

and the detail coefficients by

$$d^i_{j,k,\ell} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, \Psi^i_{j,k,\ell}(x,y)\, \mathrm{d}x\, \mathrm{d}y. \qquad (8)$$

The key property of wavelets used in this paper is the fact that the expansion (6) induces the following decomposition of $\mathcal{L}^2(\mathbb{R}^2)$

$$\mathcal{L}^2(\mathbb{R}^2) = \mathcal{V}_J \oplus \mathcal{W}^{detail}_J \oplus \mathcal{W}^{detail}_{J+1} \oplus \cdots \qquad (9)$$

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

In this paper we use Haar wavelets for reasons that will become apparent below. Each scaling function $\phi_{j,k}(x)$ and wavelet function $\psi_{j,k}(x)$ in the Haar system is supported 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 [10], [13]. Subsequently, we may associate the functions $\Phi_{j,k,\ell}$ and $\Psi^i_{j,k,\ell}$ $(i = 1,2,3)$ with the rectangular cell $c^j_{k,\ell} \triangleq I_{j,k} \times I_{j,\ell}$.

B. Wavelet decomposition of the risk measure

Without loss of generality, in the sequel we take $\mathcal{W} = [0,1] \times [0,1]$, 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 the previous discussion that the Haar wavelet decomposition of a function f defined over $\mathcal{W}$ at resolution level $J \geq J_{\min}$, given by

$$\begin{aligned} f(x,y) = &\sum_{k,\ell=0}^{2^{J_{\min}}-1} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(x,y) \\ &+ \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J-1} \sum_{k,\ell=0}^{2^j-1} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x,y) \end{aligned} \qquad (10)$$

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

Assume now that we are given a function $\mathsf{rm} : \mathcal{W} \mapsto [0,1]$ that represents the "risk measure" at the location $\mathsf{x} = (x,y)$. For instance, one may choose

$$\mathsf{rm}(\mathsf{x}) = \begin{cases} (d_{\max} - \min_{\mathsf{y} \in \mathcal{O}} \|\mathsf{x} - \mathsf{y}\|_\infty)/d_{\max}, & \text{if } \mathsf{x} \in \mathcal{F}, \\ 1, & \text{if } \mathsf{x} \in \mathcal{O}, \end{cases} \qquad (11)$$

[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 (7) and (8).

where $d_{\max} \triangleq \max_{\mathsf{x}\in\mathcal{F}} \min_{\mathsf{y}\in\mathcal{O}} \|\mathsf{x}-\mathsf{y}\|_\infty$. Alternatively, one may think of rm as the probability that $(x, y) \in \mathcal{O}$.

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

$$\mathcal{N}(\mathsf{x}_0, r_j) \triangleq \{\mathsf{x} \in \mathcal{W} : \|\mathsf{x}_0 - \mathsf{x}\|_\infty \le r_j\}. \tag{12}$$

The situation is depicted in Fig. 2. The choice of $J_{\max}$ is

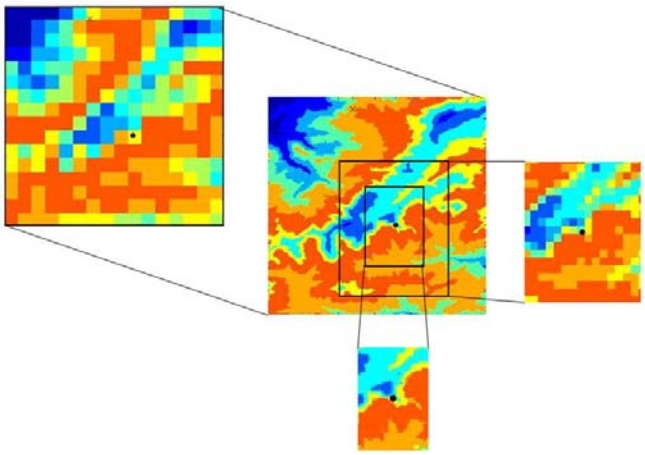


Fig. 2. Multiresolution representation of the environment according to the distance from the current location of the agent.

dictated by the requirement that at this level all cells can be resolved into either free or occupied cells. The choice of $J_{\min}$ as well as the values of r_j are typically dictated by the on-board computational resources. The choice of the norm in (12) is dictated by the use of Haar wavelet transform in the sequel.

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

$$\mathcal{K}(j) \triangleq \{k \in \mathcal{I}(j) : I_{j,k} \cap [x_0 - r(j), x_0 + r(j)] \neq \varnothing\}, \tag{13a}$$

$$\mathcal{L}(j) \triangleq \{\ell \in \mathcal{I}(j) : I_{j,\ell} \cap [y_0 - r(j), y_0 + r(j)] \neq \varnothing\}. \tag{13b}$$

The wavelet decomposition of rm, given by

$$\begin{aligned}\mathsf{rm}(x, y) = &\sum_{k,\ell\in\mathcal{I}(J_{\min})} a_{J_{\min},k,\ell}\, \Phi_{J_{\min},k,\ell}(x, y) \\ &+ \sum_{i=1}^{3} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{\substack{k\in\mathcal{K}(j)\\ \ell\in\mathcal{L}(j)}} d^i_{j,k,\ell} \Psi^i_{j,k,\ell}(x, y),\end{aligned} \tag{14}$$

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

$$\mathcal{C}_d = \Delta C_d^{J_{\min}} \oplus \cdots \oplus \Delta C_d^{J_{\max}}. \tag{15}$$

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

C. Fast lifting wavelet transform (FLWT)

Implementing the wavelet transform in practice requires dealing with a discrete signal. The most efficient implementation to date involves the use of filter banks. Figure 3 shows

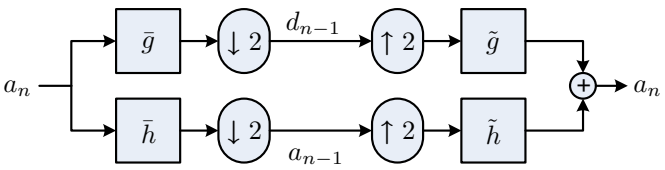


Fig. 3. A typical one-stage filter bank used to implementing the discrete wavelet transform.

a discrete signal a_n filtered by two complementary high- and low-pass (decomposition) filters $\bar{g}$ and $\bar{h}$ before it is downsampled. The result 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 the two filtered signals are added together. This results in perfect reconstruction of the original signal. Details can be found, for instance, in Refs. [14], [15].

The fast lifting wavelet scheme originally introduced in Refs. [16] and [17] is a new method for constructing wavelets directly in the time domain, avoiding the use of Fourier analysis. Moreover, the scheme can be extended to construct the so-called second generation wavelets; the latter have certain benefits for handling boundary effects, irregular samples, and arbitrary weight functions [15].

The typical lifting decomposition scheme is depicted in Fig. 4. 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 corresponding operators $\mathcal{P}$ and $\mathcal{U}$ (dual and primal lifting, or prediction and update, respectively). Finally, the results are normalized by constants k_a and k_d to end up with a coarser 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

$$d_{n-1,k} = a_{n,2k+1} - a_{n,2k}, \tag{16}$$

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

$$a_{n-1,k} = a_{n,2k} + d_{n-1,k}/2. \tag{17}$$

It has been proved that all classical wavelet transforms can be implemented using the lifting scheme [18]. 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, and

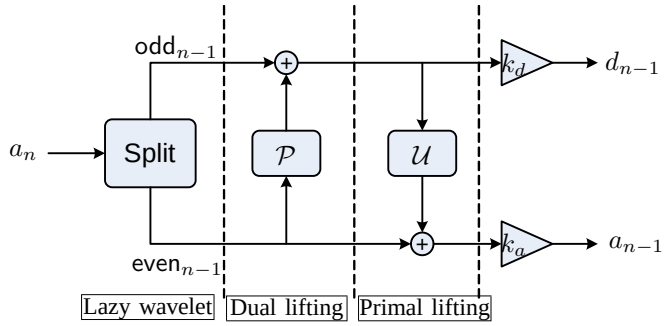


Fig. 4. One step decomposition using the lifting scheme with the lazy wavelet.

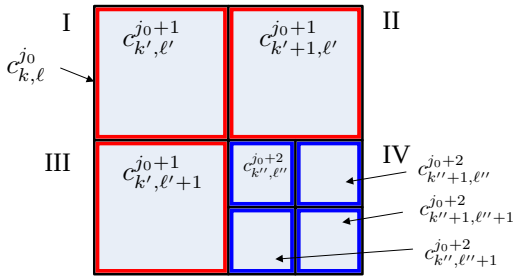


Fig. 5. Multi-resolution cell subdivision across different levels.

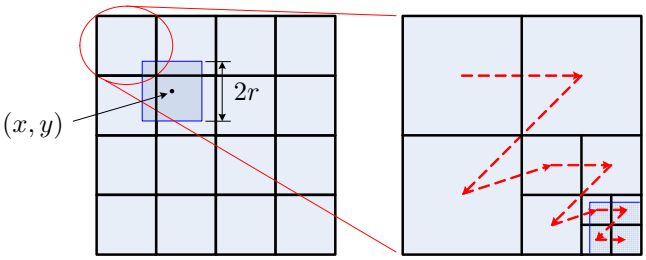


Fig. 6. Recursive raster scan for identifying independent cells.

so on. In particular, the lifting scheme fits many applications where the input data consists of integer samples, such as in image compression and processing. 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 reversible, allowing perfect reconstruction [19]. This reconstruction is possible by adopting the sequential transform [20] by modifying Eq. (17) as follows

$$\begin{aligned} d_{n-1,k} &= a_{n,2k+1} - a_{n,2k}, \\ a_{n-1,k} &= a_{n,2k} + \lfloor d_{n-1,k}/2 \rfloor. \end{aligned} \tag{18}$$

where, $\lfloor \cdot \rfloor$ is a rounding operator. In the sequel, we use the fast lifting Haar transform with integer arithmetic on each sample of integer values.

III. The Connectivity Graph

A. Computation of Adjacency List from the FLWT

To the cell decomposition (15) we assign a topological graph $\mathcal{G}$, whose nodes represent the cells $c^j_{k,\ell}$ in the decomposition (15). In this section we show that the connectivity matrix of the graph $\mathcal{G}$ can be computed efficiently directly from the wavelet coefficients of the FLWT. Equivalently, the cell decomposition associated with the wavelet coefficients can be used to construct the corresponding graph structure.

Since 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 a square cell $c^j_{k,\ell}$, 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 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 an *independent* cell if it is associated with one 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 $j_0 \le j \le J_{\max}$ are 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. 5 the 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 quadrants 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 a Haar wavelet transform of the risk measure function rm up to level $J_{\min}$. The coarsest level of the cell dimension is set to $J_{\min}$. In Fig. 6 the initial coarse grid is drawn on the left. The agent is located at $\mathsf{x} = (x, y)$ and the high resolution horizon is r. Recalling the expression (13), 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 (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 6 illustrates the recursive raster scan search. Once a cell is recognized as an independent cell, we assign a node in the graph 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 provides the necessary connectivity information between the cells.

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. Note that two adjacent cells that have been identified as independent are linked to one another, so the raster scan method searches only adjacent cells along the direction: left, top-left, top, and top-right. In addition, because we deal with cells of different dimensions, it is required to devise a generic method to find the adjacency relationship between them.

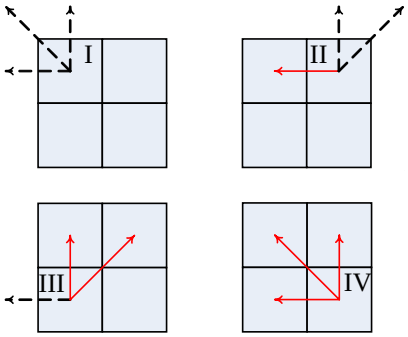


Fig. 7. Basic connectivity properties with respect to the location of the leaf cell.

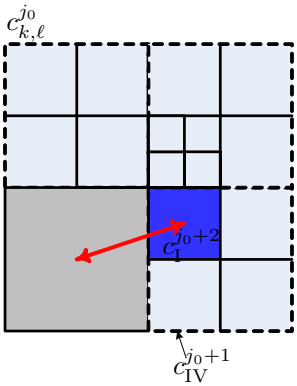
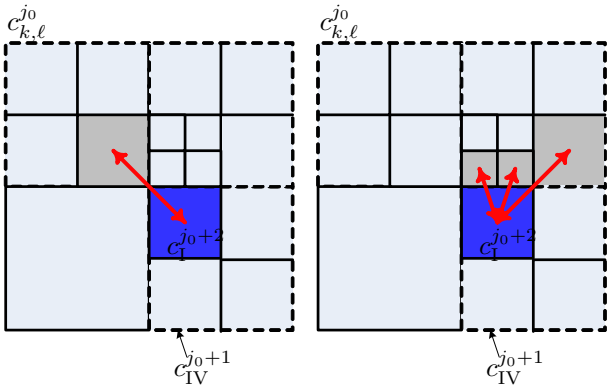


Fig. 8. Searching an adjacent cell along the left direction.



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

Fig. 9. Expanded adjacency search algorithm.

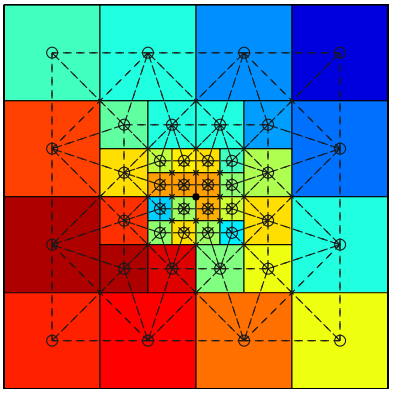


Fig. 10. Cell subdivision over different levels.

Figure 7 illustrates the basic connectivity relationship assigned to each leaf cell inside a parent cell. The dashed arrow represents the external connection from a leaf cell beyond the parent cell, and the solid arrow represents internal connection between leaf cells inside the parent cell. Both connections implicitly assume that the adjacent cell could vary in levels from that of parent cell to $J_{\max}$ (external connection), or from that of current cell to $J_{\max}$ (internal connection).

A leaf cell inherits its external connection properties from the cell which contains the leaf cell, and as a leaf cell of the top-most parent cell. Figure 8 shows this inheritance property. 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 $c_{k,\ell}^{j_0}$. The cell $c_{\mathrm{IV}}^{j_0+1}$ is located on the fourth quadrant inside the top-most parent cell $c_{k,\ell}^{j_0}$ and has the internal connectivity whose property is inherited to the cell $c_{\mathrm{I}}^{j_0+2}$ for left, top-left, and top directional search. The search along each direction will end up with the adjacency relationship between neighboring cells. As depicted in Fig. 8, the left directional search finds one adjacent cell from the current cell along the left direction by bookkeeping the associated detail coefficients of the opposite cell, which determines the level of the opposite independent cell by $j_0 + 1$.

The previous algorithm expands the search to the higher levels if the opposite cell is not an independent cell, that is, if it is comprised of finer cells. This expansion subsequently forces a search of cells of finer dimension which are, of course, neighboring to the current cell. Subsequently, the detail coefficients are examined in order to locate the next finer cell which is adjacent to the current cell. For the top-left search direction, as illustrated in Fig. 9(a), the cell at level j_0+2 located at the top-left corner of the current cell is found as an independent adjacent cell. Another example is shown in Fig. 9(b). There are two independent cells at level $j_0 + 3$ and one at level $j_0 + 2$ which are adjacent to the current cell at level $j_0 + 2$. The adjacency relationship is found accordingly, by examining and expanding the opposite cell, with the corresponding detail coefficients. Finally, Fig. 10 shows an example of a graph structure constructed from the wavelet coefficients. The dashed lines show the connectivity between the cells in the graph.

B. Cost Assignment

We associate each node v of $\mathcal{G}$ to some point $\mathsf{x} \in c_{k,\ell}^{j}$. Without loss of generality, we choose the center of the cell $c_{k,\ell}^{j}$. Let $\mathsf{cell}_{\mathcal{G}}(v)$ denote the center of the corresponding cell in this case. If $\mathsf{x} \in c_{k,\ell}^{j}$ we will write $v = \mathsf{node}_{\mathcal{G}}(\mathsf{x})$.

To each edge (u, v) of $\mathcal{G}$ we assign a cost as follows

$$\mathcal{J}(u,v) = \mathsf{rm}(\mathsf{cell}_{\mathcal{G}}(v)) + \alpha \|\mathsf{cell}_{\mathcal{G}}(u) - \mathsf{cell}_{\mathcal{G}}(v)\|_2. \quad (19)$$

where $\alpha \geq 0$ is a weight constant. The first term in (19) is proportional to the probability that the terminal node is occupied by an obstacle and 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 consecutive, adjacent nodes in $\mathcal{G}$ as follows $\mathcal{P} = (v_0, v_1, \ldots, v_q)$. We can then assign a cost to each node in the path $\mathcal{P}$, induced by the two-node transitioning cost, iteratively, via

$$\mathcal{H}(v_i) = \mathcal{H}(v_{i-1}) + \mathcal{J}(v_{i-1}, v_i), \quad i = 1, \ldots, q. \qquad (20)$$

The value of $\mathcal{H}(v_k)$ represents the (accumulated) cost of the path from v_0 to v_k $(k \leq q)$. The shortest path problem is then to find a path that minimizes the accumulated cost from the initial to the destination node, or determine that such a path does not exist.

IV. Multiresolution Path Planning

Once the adjacency list is known, 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 $\mathcal{W}$. 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 final nodes, respectively such that $v_1^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_0)$ and $v_f^0 = \mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}_f)$. Using Dijkstra's algorithm (or any other similar algorithm) we find 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 the ordered sequence of l_0 nodes as follows

$$\mathcal{P}(t_0) = (v_1^0, v_2^0, \cdots, v_{l_0-1}^0, v_{l_0}^0 = v_f^0).$$

It is assumed that v_2^0 is free owing to the high resolution decomposition of $\mathcal{W}$ close to x_0. The agent subsequently moves from v_1^0 to v_2^0. Let now t_1 be the time the agent is at the location $\mathsf{x}(t_1) = \mathsf{cell}_{\mathcal{G}(t_0)}(v_2^0)$ and let $\mathcal{C}_d(t_1)$ be the multiresolution cell decomposition of $\mathcal{W}$ around $\mathsf{x}(t_1)$ with corresponding topological graph $\mathcal{G}(t_1)$. Applying again Dijkstra's algorithm we find a (perhaps new) path in $\mathcal{G}(t_1)$ from $v_1^1 = \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1))$ 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 nodes as follows

$$\mathcal{P}(t_0) = (v_1^1, v_2^1, \cdots, v_{l_1-1}^1, v_{l_1}^1 = v_f^1).$$

The agent subsequently moves to $\mathsf{x}(t_2) = \mathsf{cell}_{\mathcal{G}(t_1)}(v_2^1)$ 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 corresponding graph $\mathcal{G}(t_i)$. Dijkstra's algorithm yields a path $\mathcal{P}(t_i)$ in $\mathcal{G}(t_i)$ of mixed and free noes of length l_i,

$$\mathcal{P}(t_i) = (v_1^i, v_2^i, \cdots, v_{l_i-1}^i, v_{l_i}^i = v_f^i),$$

where $v_1^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}(t_i))$ and $v_f^i = \mathsf{node}_{\mathcal{G}(t_i)}(\mathsf{x}_f)$ if such a path exists. The process is continued until some time t_f when $\|\mathsf{x}(t_f) - \mathsf{x}_f\| < 1/2^{J_{\max}}$, at which time the algorithm terminates. At the last step the agent moves from $\mathsf{x}(t_f)$ to x_f.

Note that the actual path $\mathsf{x}(t_0), \mathsf{x}(t_1), \ldots, \mathsf{x}(t_f)$ followed by the agent is given by the sequence of nodes $\mathsf{node}_{\mathcal{G}(t_0)}(\mathsf{x}(t_0)), \mathsf{node}_{\mathcal{G}(t_1)}(\mathsf{x}(t_1)), \ldots, \mathsf{node}_{\mathcal{G}(t_f)}(\mathsf{x}(t_f))$.

Several improvements and refinements of the previous baseline path-planning algorithm are possible. First, for dynamically changing environments the use of D^* in lieu of Dijkstra's algorithm should speed up the calculation of the shortest path. Second, we can postpone the calculation of the time-consuming path search over $\mathcal{G}(t_i)$ until the agent visits node v_{1+q}^i where $q > 1$. This is especially true if the node v_{1+r}^i corresponds to the finest resolution scale in the cell decomposition at time t_i. By construction, all nodes $v_1^i, v_2^i, \ldots, v_{r-1}^i$ in the path $\mathcal{P}(t_i)$ also represent cells in the finest resolution and hence the path from v_1^i to v_r^i is composed only of free cells.

V. Simulation Results

In this section we present simulation results of the proposed algorithm for a non-trivial scenario. The environment is assumed to be square of dimension 512×512 units. Hence $N = 9$ is the finest resolution possible. For simplicity, only two levels of resolution have been chosen to represent the environment. Inside an area of 100×100 unit cells we employ a high resolution approximation and outside this area we employ a low resolution approximation of $\mathcal{W}$.

The environment $\mathcal{W}$ is an actual topographic (elevation) map of a certain US state with fractal-like characteristics, shown in Fig. 11. The initial and final positions of the agent are also shown in this figure. The objective is for the agent (e.g., a UAV) to follow a path from A to B while flying as low as possible, and below a certain elevation threshold. Areas with bright colors in Fig. 11 correspond to areas of low risk (elevation in this case) and darker colors correspond to areas of high risk (elevation in this case) that should be avoided. Solving the path-planning problem on-line at this resolution is computationally prohibitive.

The results from the multiresolution path-planning algorithm using a fine resolution level $J_{\max} = 5$, and a low resolution at level $J_{\min} = 3$ are shown in Fig. 12. Specifically, Fig. 12 shows the evolution of the path at different time steps as the agent moves to the final destination. Figure 12(a) shows the agent's position at time step $t = t_{15}$ along with the best proposed path to the final destination at that time. Similarly, Fig. 12(b) shows the agent's position at time step $t = t_{50}$ along with the best proposed path to the final destination at that time. As seen in Fig. 12(c), the actual path followed by the agent differs significantly from the one predicted in either Figs. 12(a) or 12(b). This is due to the fact that at time t_{15} and t_{50} the agent does not have complete information outside the high resolution zone, and the predicted path actually penetrates the obstacle space $\mathcal{O}$. At time t_{50}, for example, the agent – being far from any obstacle – fails to anticipate the upcoming collision. As the agent gets closer to the obstacle however, and new information is gathered, the existence of the obstacle forces the agent to redirect its path. The agent reaches the final destination x_f in a collision free manner, as seen in Fig. 12(c). The actual path followed lies completely inside areas of low elevation level, which verifies the collision-free nature of the path.

VI. Experimental Test-Bed

A UAV platform based on the airframe of an off-the-self R/C model airplane (see Fig. 1) has been developed, and is used to implement the hierarchical, wavelet-based path-planning algorithm described above. The development of the hardware and software was done completely in-house

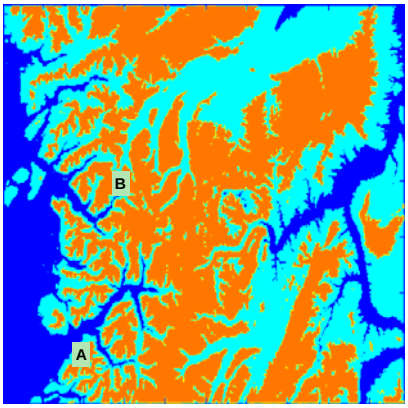
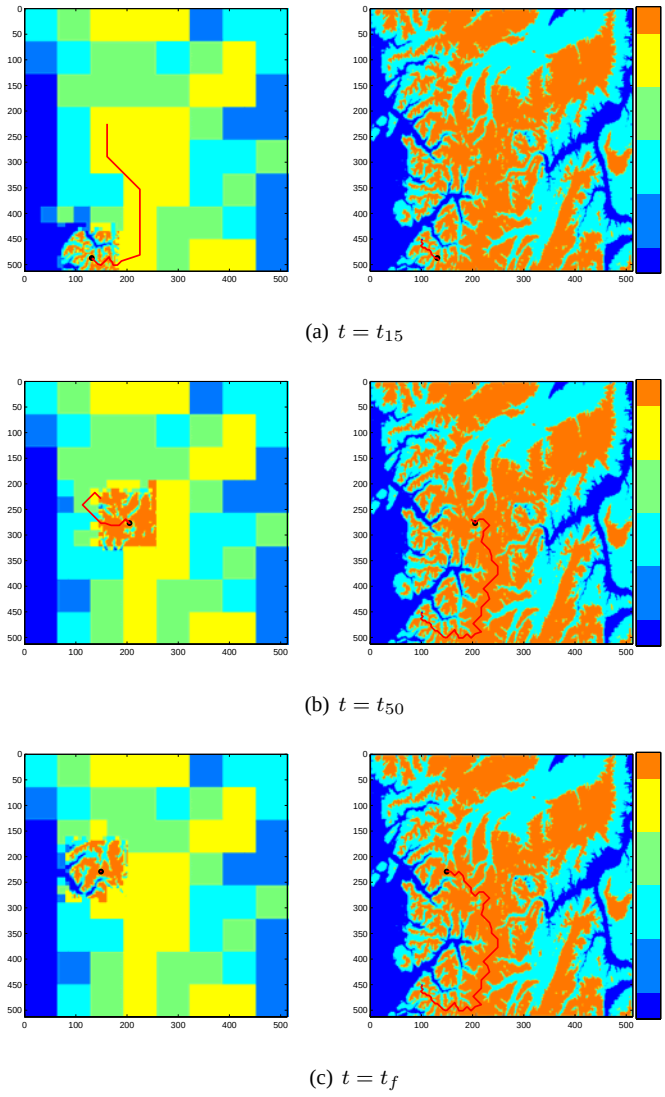


Fig. 11. Plot of risk measure (elevation) for the whole configuration space using a 512×512 unit cell resolution. The blue color corresponds to areas of obstacles. The initial configuration of the agent is denoted by A and the desired final configuration is denoted by B.

by Georgia Tech graduate and undergraduate students. The overall architecture of the UAV system is shown in Fig. 13. The main subsystems are the UAV autopilot, the ground station, and the interconnection between the two. The on-board autopilot is equipped with a microcontroller, sensors and actuators, along with the communication devices that allow full functionality for autonomous control. The micro-controller (Rabbit RCM-3400 at 30 MHz, 512 KB RAM) provides data acquisition, processing, and communication with the ground station. The microcontroller board is shown in Fig. 1. It also runs the main control software. The on-board sensors include three-axis angular rate sensors, three-axis accelerometers, a three-axis magnetic compass, a GPS sensor, an engine RPM sensor, absolute and differential pressure sensors, battery voltage, fuel level and temperature sensors.

The ground station consists of a laptop computer with a wireless communication modem. The laptop runs a Windows-based Graphical User Interface (GUI) program, shown in Fig. 14. The ground station program provides real-time flight information by displaying all relevant system parameters, sensor readings, etc. A graphical dashboard representing a virtual horizon, altitude and speed has been adopted in the GUI panel to show graphically all relevant information. A map of the area of the UAV's operation can be overlaid on the map panel in order to provide the user with the navigational details of the airplane via GPS data.

A hardware-in-the-loop simulation (HILS) environment has also been developed to validate the UAV autopilot hardware and software development. A full 6-dof nonlinear dynamic model is used in conjunction with a linear approximation of the aerodynamic forces and moments, along with realistic Earth gravitational (WGS-84) and magnetic field models. Detailed models for the sensors and actuators have also been incorporated, along with the capability of real-time 3-D visualization. An R/C transmitter is connected to the 6-dof simulator program emulating remote pilot stick commands. Six channel commands are recorded: four chan-



(a) $t = t_{15}$

(b) $t = t_{50}$

(c) $t = t_f$

Fig. 12. Path evolution and replanning. Figures on the left show the currently tentative optimal path obtained from Dijkstra's 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.

nels for control surface commands ($\delta_a, \delta_e, \delta_t$ and δ_r) and two auxiliary commands for toggling between autonomous control mode and remote pilot mode. Four independent computer systems are used in the hardware-in-the-loop simulation (HILS) as illustrated in Fig. 15: a 6-dof simulator, the flight visualization computer, the autopilot microcontroller, and the ground station computer console. Further details on the UAV platform, autopilot and HILS set-up can be found in [21], [22] and [23].

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, multi-resolution path

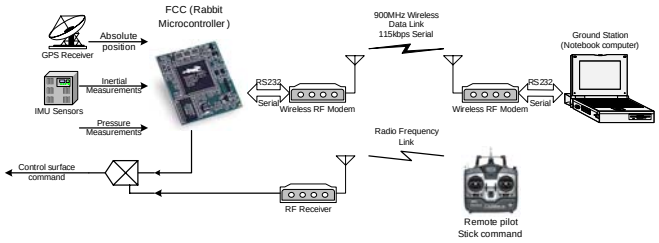


Fig. 13. Hardware system configuration schematic of the UAV test bed. All hardware integration, electronics and associated software was developed in-house by Georgia Tech students.

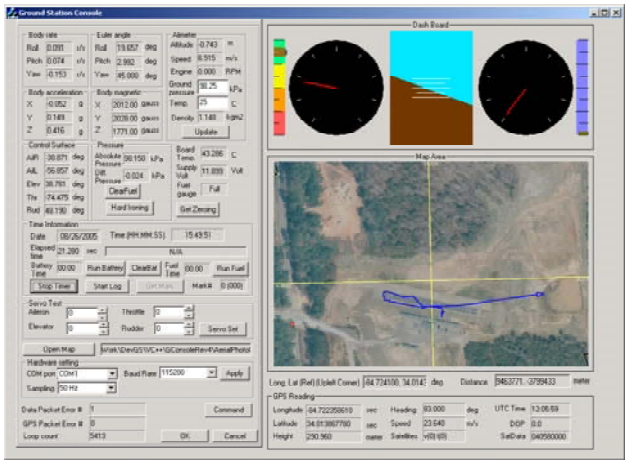


Fig. 14. The ground station GUI program.

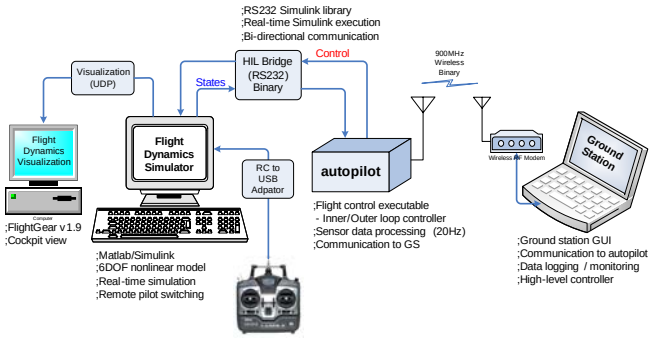


Fig. 15. High fidelity hardware-in-the-loop simulation (HILS) environment that enables rapid testing of the designed UAV control laws.

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 adjacency matrix 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. Several extensions of the baseline methodology presented here are possible, and will be addressed in the future.

Acknowledgement: This work has been supported in part by NSF (award no. CMS-0510259) and ARO (award no. W911NF-05-1-0331). The author would also like to acknowledge the help of Dongwon Jung, Efstathios Bakolas and Raghvendra Cowlagi in this work.

REFERENCES

[1] R. A. Brooks and T. Lozano-Perez, "A subdivision algorithm on configuration space for findpath with rotation," in *IEEE Trans. Syst., Man, Cybern.*, vol. 15, pp. 224–233, 1985.

[2] D. Zhu and J. Latombe, "New heuristic algorithms for efficient hierarchical path planning," in *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 9–20, 1991.

[3] H. Noborio, T. Naniwa, and S. Arimoto, "A quadtree-based path-planning algorithm for a mobile robot," *Journal of Robotic Systems*, vol. 7, no. 44, pp. 555–574, 1990.

[4] D. Godbole, T. Samad, and V. Gopal, "Active multi-model control for dynamic maneuver optimization of unmanned air vehicles," in *Proceedings of of the IEEE International Conference of Robotics and Automation*, (San Francisco, CA), pp. 1257–1262, 2000.

[5] D. K. Pai and L. M. Reissell, "Multiresolution rough terrain motion planning," in *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 19–33, 1998.

[6] S. Kambhampati and L. S. Davis, "Multiresolution path planning for mobile robots planning for mobile robots," *IEEE Journal of Robotics and Automation*, pp. 135–145, 1986.

[7] S. Behnke, "Local multiresolution path planning," *Lecture Notes in Computer Science*, vol. 3020, pp. 332–343, 2004.

[8] I. Daubechies and W. Sweldens, "Factoring wavelets transforms into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, 1998.

[9] T. Lozano-Perez and M. A. Wesley, "Automatic planning for planning collision-free paths among polyhedral obstacles," in *IEEE Trans. Syst., Man, Cybern.*, vol. 11, pp. 681–698, 1981.

[10] C. S. Burrus, R. A. Gopinath, and H. Guo, *Introduction to Wavelets and Wavelet Transforms*. New Jersey, NJ: Prentice Hall, 1998.

[11] S. G. Mallat, "A theory for multiresolution signal decomposition, the wavelet representation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, 1989.

[12] A. Cohen, *Numerical Analysis of Wavelet Methods*, vol. 32. Amsterdam: Elsever Science, 2003.

[13] D. F. Walnut, *An Introduction to Wavelet Analysis*, vol. 32. Boston: Birkhauser, 2003.

[14] D. L. Donoho, "Smooth Wavelet Decompositions with Blocky Coefficient Kernels," in *Recent Advances in Wavelet Analysis*, pp. 1–43, Academic Press, 1993.

[15] W. Sweldens and P. Schröder, "Building Your Own Wavelets at Home," in *Wavelets in Computer Graphics*, pp. 15–87, ACM SIGGRAPH Course notes, 1996.

[16] W. Sweldens, "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," in *Wavelet Applications in Signal and Image Processing III*, pp. 68–79, 1995.

[17] W. Sweldens, "The Lifting Scheme: A Construction of Second Generation Wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1997.

[18] I. Daubechies and W. Sweldens, "Factoring Wavelet Transform into Lifting Steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[19] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet Transforms That Map Integers to Integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, 1998.

[20] V. K. Heer and H.-E. Reinfelder, "A Comparison of Reversible Methods for Data Compression," in *Medical Imaging IV: Image Processing*, vol. Proc. SPIE Vol. 1233, pp. 354–365, 1990.

[21] D. Jung, D. Zhou, R. Fink, T. Williams, J. Moshe, and P. Tsiotras, "Design and development of a low-cost test-bed for undergraduate education in UAVs," in *44th IEEE Conference on Decision and Control/European Control Conference ECC 2005*, (Seville, Spain), pp. 2739–2744, Dec. 12–15 2005.

[22] D. Jung and P. Tsiotras, "Inertial attitude and position reference system development for a small UAV," in *AIAA Infotech at Aerospace*, (Rohnert Park, CA), May 7–10 2007. AIAA Paper 07-2763.

[23] D. Jung and P. Tsiotras, "Modelling and hardware-in-the-loop simulation for a small unmanned aerial vehicle," in *AIAA Infotech at Aerospace*, (Rohnert Park, CA), May 7–10 2007. AIAA Paper 07-2768.