Multiresolution Analysis on Graphs for Computationally Efficient Path Planning

Aniket Bhatia
IIT Bombay
aniketb@iitb.ac.in

Sucheta Ravikanti
IIT Bombay
160040100@iitb.ac.in

Abstract

In many robotics applications such as drone delivery, it is of interest to develop path planning algorithms that are compute and memory efficient, allowing for cheap onboard implementation using limited resources. In this project we explore the use of multiresolution path planning to address this problem. First, we successfully implemented a previously proposed approach that employs distance-dependent downsampling via 2D Haar-wavelet decomposition, followed by Dijkstra's algorithm for subsequent path planning on the downsampled graph. Next, we explored variants of this scheme by using Gaussian and Daubechies scaling functions. The algorithms discussed so far only work for maps that are defined on a regular 2D grid, with the corresponding Euclidean metric and uniform topology greatly expediting algorithm design. Finally, we relax this requirement for a grid-like graph and consider the much harder problem of multiresolution path planning on arbitrary graphs. To do this, we use ideas from Graph Signal Processing to define notions of frequency, wavelets, filtering, and downsampling on arbitrary graphs. To the best of our knowledge, our proposed algorithm is the first instance of multiresolution path planning on arbitrary graphs.

1. Multiresolution Path Planning on Grids

1.1. Topology of the Problem

We assume that our bot is a point-mass that can move in a square region $R = [0, 2^D] \times [0, 2^D]$. Furthermore, there exists an intensity map $F : R \to \mathbb{R}_+$, which can correspond to say the altitude at each point on the map. This intensity map is known at a finite resolution $j_{max} > -D$ meaning that the function F is piecewise constant over each of the square cells $[2^{-j_{max}}k, 2^{-j_{max}}(k+1)] \times [2^{-j_{max}}l, 2^{-j_{max}}(l+1)]\,; k, l = 0, 1, ..., 2^{D-j_{max}-1}$. Thus the intensity map can be thought of as being discretely defined on square pixels.

The cell decomposition C of a map is a finite collection of disjoint, convex subsets C_n of R called cells, such that $\bigcup_{n=1}^{N_c} C_n = R$ and F is constant over the cell $C_n\ \forall n = 1, 2, ..., N_c$, where $N_c \in \mathbb{N}$. In this section, as we are working with grid-like graphs, we consider square cells and use $C(j, k, l)$ to denote a cell of size 2^{-j} with its center at $(2^{-j}k + 2^{-j-1}, 2^{-j}l + 2^{-j-1})$. The collection of cells $C_j := C(j, k, l);\ k, l = 0, 1, ..., 2^D - 1$ is a uniform cell decomposition at resolution level j. We associate with the highest resolution cell decomposition $C_{j_{max}}$ a graph $\bar{G} = (\bar{V}, \bar{E})$, such that each node in $\bar{V}$ corresponds to a unique cell in $C_{j_{max}}$. Two nodes in $\bar{V}$ are adjacent if the corresponding cells in $C_{j_{max}}$ are geometrically adjacent, wherein cells with two vertices in common are adjacent. The edge set $\bar{E}$ consists of all pairs $(\bar{u}, \bar{v})$, where $\bar{u}, \bar{v} \in \bar{V}$ are adjacent nodes.

We introduce (and will later explicitly define) an edge cost function $\bar{g} : \bar{E} \to \mathbb{R}_+$, which assigns to each edge of $\bar{G}$ a non-negative cost of transitioning across this edge. For initial and final nodes $\bar{u}_i, \bar{u}_f \in \bar{V}$, a path $\bar{\pi}(\bar{u}_i, \bar{u}_f) := (\bar{u}_0, \bar{u}_1, ..., \bar{u}_{\bar{P}})$ in $\bar{G}$ such that $\bar{u}_k^{\bar{\pi}} \in \bar{V}$, $(\bar{u}_{k-1}^{\bar{\pi}}, \bar{u}_k^{\bar{\pi}}) \in \bar{E}$, $k = 1, ..., \bar{P}$, with $\bar{u}_0^{\bar{\pi}} = \bar{u}_i$ and $\bar{u}_{\bar{P}}^{\bar{\pi}} = \bar{u}_f$, and $\bar{u}_p^{\bar{\pi}} \neq \bar{u}_r^{\bar{\pi}}$ for $p, r \in 0, 1, ..., \bar{P}$. For brevity, and when there is no ambiguity, henceforth we will supress arguments in $\bar{\pi}$. The cost of a path $\bar{\pi}$ is defined as $\bar{J}(\bar{\pi}) := \sum_{k=1}^{\bar{P}} \bar{g}((\bar{u}_{k-1}^{\bar{\pi}}, \bar{u}_k^{\bar{\pi}}))$. The path planning problem is to find a path $\bar{J}(\bar{\pi}^*) \leq \bar{J}(\bar{\pi})$ for every path $\bar{\pi}$ in $\bar{G}$.

1.2. Multi-resolution Path Planning

Dijkstra's algorithm [11] is an efficient dynamic programming algorithm to solve such a path planning problem, with a runtime complexity of $\Theta(|V| + |E|log|V|)$. However, the graph $\bar{G}$ associated with large environments can consist of a very large number of nodes and edges. For practical, real-time applications, the execution of Dijkstra's algorithm may thus be compute and memory intensive and thus it is desirable to work with a coarser approximation for such large graphs. Another important observation is that the bot needs to know the exact path only for the next few time-

steps at any point in time, allowing for the exact path to be computed iteratively. Put together, this hints at the possibility of constructing a coarser graph at each timestep that has far fewer nodes and edges, while retaining sufficient information to accurately plan the next few steps of the desired path.

The idea of multiresolution path planning for grid-like maps was proposed in [9]. Subsequently, theoretical analysis and a more efficient implementation was discussed in [1]. This algorithm was applied to radial maps by converting them to rectangular maps via a conformal mapping in [10]. All these papers use the Haar multiresolution for downsampling. Informally, the algorithms in these papers operate as follows: at each timestep a multiresolution cell decomposition is constructed. This decomposition retains high resolution in the immediate vicinity of the bot's current location and approximates the environment in regions farther away.

These algorithms thus construct a smaller graph G from $\bar{G}$ at each timestep. We will refer to a multiresolution approximation of the intensty map F as F^{mr}. Each F^{mr} uniquely corresponds to a cell decomposition C^{mr} that consists of cells of different sizes. We associate with C^{mr} the graph $G = (V, E)$. Each node $u \in V$ corresponds to a set $W(u, V) := \bar{w}_0^u, \bar{w}_1^u, ..., \bar{w}_{S(u)}^u \subset \bar{V}$, such that $W(u, V)_{u \in V}$ is a partition of $\bar{V}$. Here $S(u)$ is the cardinality of $W(u, V)$. Thus the multiresolution cell decomposition graph G approximates the graph $\bar{G}$ by representing each set of nodes $W(u, V) \subset \bar{V}$ with a single node $u \in V$. Two nodes $u, v \in V$ are adjacent in G, i.e $(u, v) \in E$ if and only if there exist $\bar{u} \in W(u, V)$ and $\bar{v} \in W(v, V)$ such that $(\bar{u}, \bar{v}) \in \bar{E}$.

The next step is to specify C^{mr} at each timestep. Let the position of the bot at time t be $p_t = (x_t, y_t)$. The neighbourhood of radius r around p_t for grid-like graphs is defined as $N(p_t, r) := x \in R : ||p_t - x||_\infty \leq r$. Having defined a radius, we can also define an annulus (region enclosed between two radii). In particular, as we wish to reduce the resolution as we move away from p_t, we define increasingly wider annular regions. The j'th annular region denoted by $A(p_t, j)$ is composed of the region between radii r_j and r_{j+1} and we choose $r_{j+1} - r_j = 1/2^{j_{max}-j}$, in line with the dyadic nature of our multiresolution scheme. Furthermore, let the scaling function for our chosen wavelet be $\phi_j(x) = 2^{j/2}\phi(2^j x)$. We proceed to convolve $A(p_t, j)$ with $\phi_j(x)$ to generate a smoothed signal, and then subsequently downsample $A(p_t, j)$ by a factor of $2^{j-j_{max}}$. Given the grid topology, this downsampling is easily done by dividing $A(p_t, j)$ into squares of side $2^{j-j_{max}}$ and retaining only the midpoint of each such square. Thus informally, we replace a square grid of pixels by a single larger superpixel.

To summarize, we partition the map into annular regions of increasing width, convolve the increasingly wider annulii with increasingly dilated versions of the wavelet scaling function, and then downsample these annular regions by increasingly aggressive downsampling factors. Indeed, the increase is exponential, leading to an (asymptotically) exponential decrease in the number of nodes and edges in going from $\bar{G}$ to G, leading to a large speedup in Dijkstra's algorithm. It can thus be inferred that multiresolution path planning is particularly desirable while working with very large $\bar{G}$, where the cost of constructing G at each timestep is outweighed by the savings in executing Dijkstra's algorithm.

1.3. Cost Assignment

A final detail that we need to specify is the choice of cost function $g(u, v)$ along the edges. We choose $g(u, v) = F^{mr}(v) \cdot S(v) + \alpha \cdot ||u - v||_2$ where α is a scalar weight and as defined before, $S(v)$ is the cardinality of the set $W(v, V)$. Note that $S(v)$ is a novel term that we have introduced in the cost function and we found that it greatly improves path planning performance. The rationale for the inclusion of this term is that the intensity value for a superpixel should be multiplied by its size to infer the total intensity cost that will be incurred while traversing *inside* this superpixel on the original map of smaller pixels. Thus the cost of transitioning from node u to node v is the weighted sum of the total intensity in v and the Euclidean distance between nodes u and v (which is the distance between the centers of the corresponding superpixels). This cost function is easily interpreted if the intensity corresponds to altitude, in which case the bot is attempting to minimize the altitude along its path (say to to maximize fuel efficiency) while simultaneously minimizing the length of the path (for faster commute time along with fuel savings).

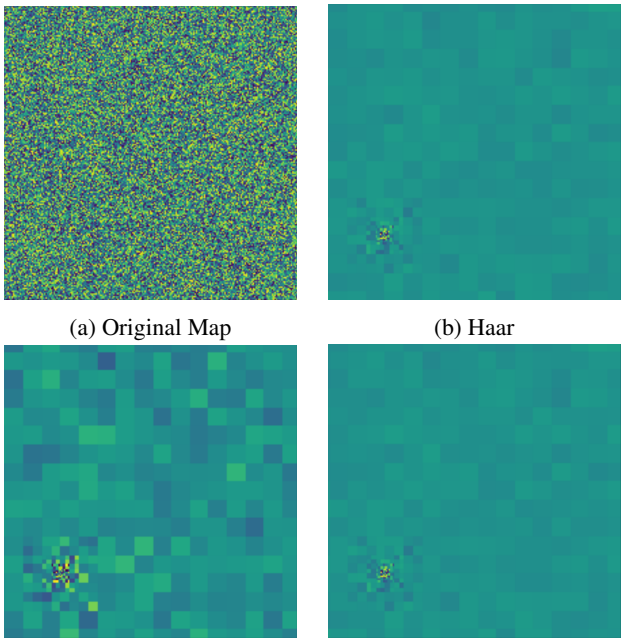
1.4. Simulation Results

Fig. 1 shows a sample map along with the corresponding multiresolution maps, constructed using different scaling functions. The center of the map is the same in all cases.

Fig. 2 shows the output path from the multiresolution path planning algorithm for different choices of scaling function. The initial and final points are the same in all cases. Note that the blue, yellow and red regions respectively have the lowest, intermediate and highest altitudes. It is thus desirable to avoid yellow and red regions as far as possible while also picking a short overall path. The bot's path is depicted by the solid black line.

2. Multiresolution Path Planning on Arbitrary Graphs

Having discussed multiresolution path planning on grids, we now proceed to design an analogous scheme for arbitrary graphs. Indeed, rapid graph traversal is a problem



(a) Original Map (b) Haar

(c) Daubechies (d) Gaussian

Figure 1: Generated multiresolution maps for different choices of scaling function.

of general theoretical and practical interest. Furthermore, graphs can be used as sparse data structures to store maps, and importantly, multimodal maps often do not conform to a grid-like topology and can have arbitrary links between nodes. A good example of this is a map that includes flights and subways apart from conventional roadways. Such maps can also be sparsified by only retaining cities and towns while excluding uninhabitable regions and water bodies, thus leading to sparse and arbitrarily linked graphs. To do this, we draw upon ideas from Graph Signal Processing (GSP) to define notions of the Fourier Transform, convolutions and filtering, wavelets and downsampling on arbitrary graphs [5].

2.1. Topology of the Problem

Consider an undirected, connected, weighted graph (if the graph consists of multiple disconnected subgraphs, then each such subgraph can be processed independently). Denote this graph by $G = \{V, E, W\}$ where V is the finite set of vertices with $|V| = N$, E the set of edges and W is the corresponding weighted adjacency matrix. If there is an edge $e = (i.j)$ connecting vertices i and j, the entry $W_{i,j}$ equals the weight of the corresponding edge, else $W_{i,j} = 0$. A signal or function $f : V \to \mathbb{R}$ defined on the vertices of the graph may be represented as a vector $f \in \mathbb{R}^N$.

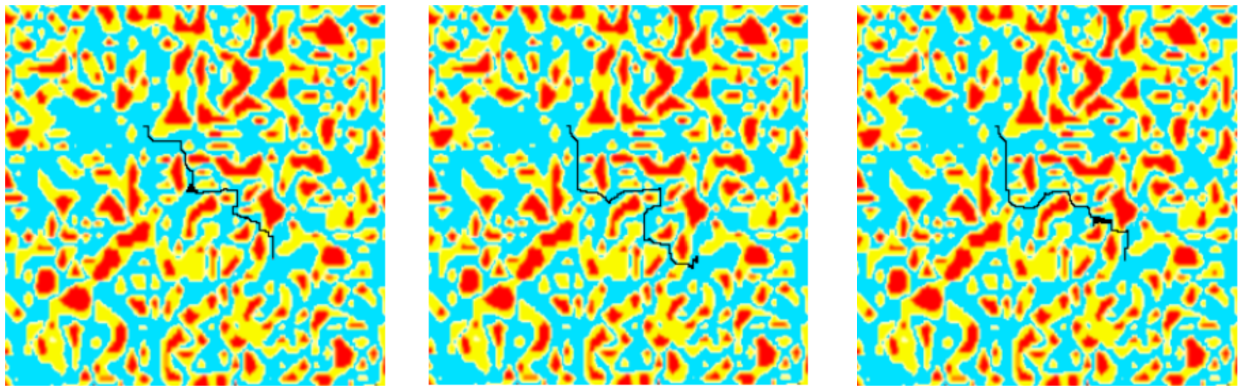
2.2. Graph Signal Processing

The non-normalized graph Laplacian, also called the combinatorial graph Laplacian is defined as $L := D - W$, where the degree matrix D is a diagonal matrix whose i'th diagonal element d_i is equal to the sum of the weights of all the edges incident to vertex i. We then have $(Lf)[i] = \sum_{j \in N_i} W_{i,j}[f(i) - f(j)]$, where the neighborhood N_i is the set of vertices connected to vertex i by an edge. As with the grids in the previous section, we can also define the notion of a radius: let $N(i, k)$ denote the set of vertices connected to vertex i by a path of k or fewer edges. In turn, an annular region $A(i, r_1, r_2)$ then comprises the set of nodes that are in $N(i, r_2)$ but not in $N(i, r_1)$, with $r_2 > r_1$. As we wish to define a dyadic multiresolution, we can sequence the radii and choose them such that $r_{j+1} - r_j = 2^j$.

Because the graph Laplcian L is a real symmetric matrix, it has a complete set of orthonormal eigenvectors, which we denote by $\{u_l\}$ with $l = 0, 1, ..., N-1$. These eigenvectors have associated eigenvalues $\{\lambda_l\}$ satisfying $Lu_l = \lambda_l u_l$. Zero appears as an eigenvalue with multiplicity equal to the number of connected components of the graph and as we are considering connected graphs, we can order the eigenvalues as $0 = \lambda_0 < \lambda_1 \leq \lambda_2 ... \leq \lambda_{N-1} := \lambda_{max}$. The Graph Fourier Transform (GFT) is then defined as $\hat{f}(\lambda_l) =< f, u_l >= \sum_{i=0}^{N-1} f(i) u_l^*(i)$. The Inverse Graph Fourier Transform (IGFT) is given as $f(i) = \sum_{l=0}^{N-1} \hat{f}(\lambda_l) u_l(i)$. The classic Fourier spectrum has the property that signals with higher frequency oscillate faster, and this property is also found to empirically hold for the GFT: eigenvectors associated with larger eigenvalues are more likely to have dissimilar values on vertices connected by an edge with high weight. Conversely, eigenvectors corresponding to smaller eigenvalues tend to be smoother in that if two vertices are connected by an edge with a large weight the values of the eigenvector at those locations are likely to be similar. This notion can be formalized by defining zero crossings as the set of edges connecting a vertex with a positive signal to a vertex with a negative signal: $Z_G(f) := \{(i, j) \in E : f(i)f(j) < 0\}$. And $|Z_G(u_l)|$ is empirically found to be a non-decreasing function of l, in line with the notion that eigenvalues associated with larger eigenvalues oscillate more.

We can now define filtering in the graph frequency domain as $\hat{f}_{out}(\lambda_l) = \hat{f}_{in}(\lambda_l)\hat{h}(\lambda_l)$, where h is a filter, f_{in} and f_{out} are respectively the input and output signals. Using the IGFT, this multiplication in the graph frequency domain then allows us to define convolution in the vertex domain as $f_{out}(i) = \sum_{l=0}^{N-1} \hat{f}_{in}(\lambda_l)\hat{h}(\lambda_l)u_l(i)$.

As with filtering, it is easier to specify wavelets in the graph frequency domain. We know that classic wavelets are bandpass filters with a scale parameter that simulaneously



(a) Haar (b) Gaussian (c) Daubechies

Figure 2: Bot's path for multiresolution path planning with different scaling functions.

increases or decreases the center frequency and bandwidth of the filter. We can thus define a collection of such bandpass filters at a set of dyadically varying scales to define a Graph Wavelet Filter Bank. The IGFT can then be used to instantiate them in the vertex domain. We denote by $\psi(j)$ the graph wavelet at resolution j. These graph wavelets can subsequently be visualized by convolving them with a delta function defined on the underlying graph. Note that the delta function centered at some node i is given as $\delta(i) = 1$, $\delta(k) = 0\ \forall k \neq i$. We can similarly define the corresponding scaling functions for these wavelets in the graph frequency domain and denote them by $\phi(j)$.

The graph frequency spectrum for Meyer wavelets is depicted in Fig. 3. Note the bandpass nature for all these graph filters. We then visualize this bank of Meyer wavelets on a 1D ring in Fig. 4. Similarly, the graph frequency spectrum for classic Mexican hat wavelets at multiple resolutions is depicted in Fig. 5. We then visualize this wavelet bank on a 2D torus in Fig. 6.

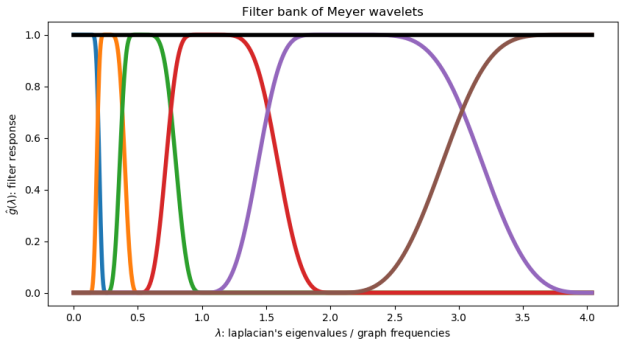
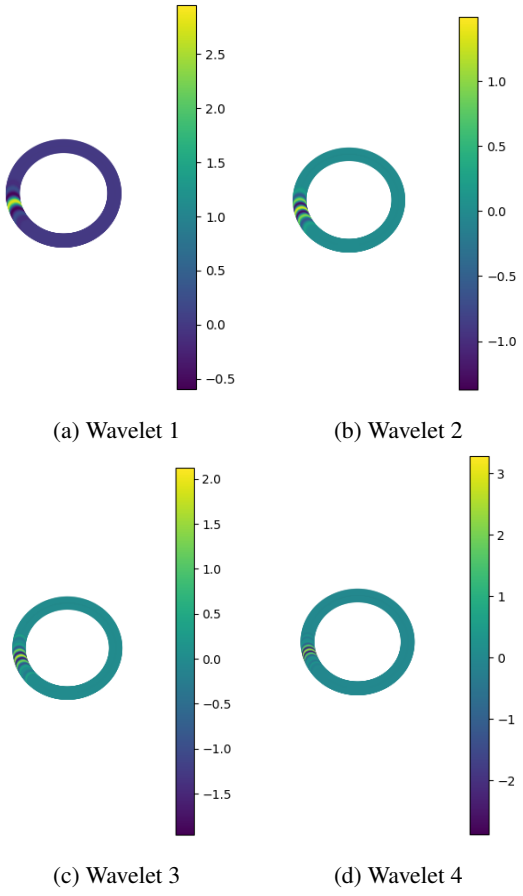


Figure 3: Graph Frequency Spectrum for Meyer wavelets.



(a) Wavelet 1 (b) Wavelet 2

(c) Wavelet 3 (d) Wavelet 4

Figure 4: Bank of Meyer wavelets visualized on a one-dimensional ring.

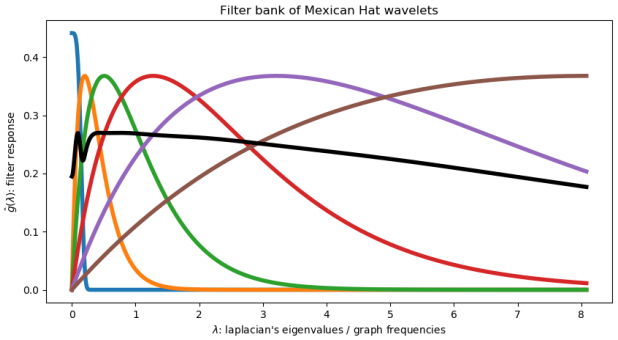
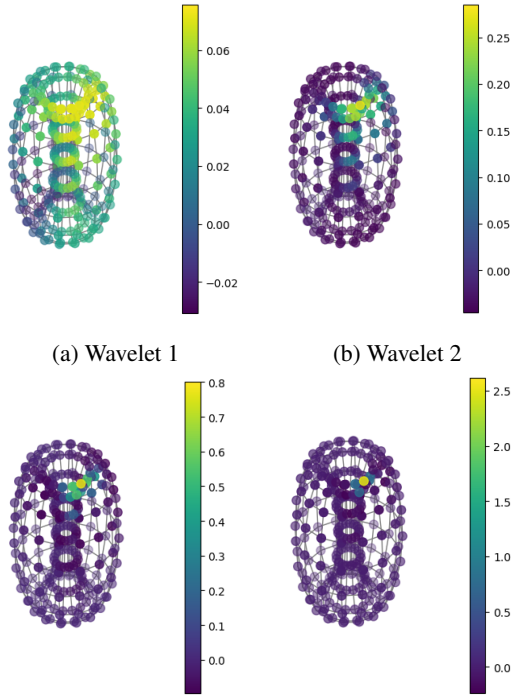


Figure 5: Graph Frequency Spectrum for Mexican hat wavelets.



(a) Wavelet 1 (b) Wavelet 2

(c) Wavelet 3 (d) Wavelet 4

Figure 6: Bank of Mexican hat wavelets visualized on a two-dimensional torus.

2.3. Downsampling

We now discuss downsampling, which is a more difficult notion to define for arbitrary graphs due to the following reasons: (i) Downsampling on regular grids entails retaining *every other* node. However, given the arbitrary topology of graphs, it is unclear how to define *every other* node. (ii) After downsampling a regular grid, the resultant graph is also a regular grid with the same topology, thus vastly simplifying the process of defining edges on the new grid. However, it is unclear how to specify the topology of the downsampled graph (and what topological features to retain), which poses the related challenge of defining the set of edges on the downsampled graph.

The downsampling method that we implemented is based on the scheme proposed in [4] that constructs a Laplacian Pyramid of Graphs. This scheme has three steps: (i) vertex selection - selecting the nodes that will be retained in the downsampled graph (ii) graph reduction - constructing the new reduced graph on the selected set of nodes (iii) graph sparsification - the previous graph reduction step tends to generate very dense graphs and it is thus desirable to sparsify this reduced graph.

The vertices that are retained are chosen to be the ones on which the eigenvector corresponding to the largest eigenvalue has a positive sign, as it is empirically observed that this eigenvector is positive on roughly half the vertices. Formally stated, given the set of vertices V, we retain the subset V_1 given as $V_1 = V_+ := \{i \in V : u_{max} \geq 0\}$. Note that it is equally acceptable to retain the subset V_-. It is also known that this eigenvector method of picking nodes has the desirable property of being exact for bipartite graphs. Examples of bipartite graphs include rings, grids, and trees. Thus in particular, this more general vertex selection method is consistent with the specific case of grids discussed in the previous section.

Once these vertices have been selected, the second step is to construct the reduced graph for the chosen nodes. The graph Laplacian of the downsampled graph is constructed from the graph Laplacian of the original graph using a method called Kron reduction. Given a graph $G = \{V, E, W\}$, we construct a reduced graph $G^{Kron} = \{V, E^{Kron}, W^{Kron}\}$ with $V_1 \subset V$. If L is the graph Laplacian of G, then the Kron reduction of L with respect to V_1, denoted by $K(L, V_1)$, is the Schur complement of L relative to $L_{V_1^c, V_1^c}$ where $L_{A,B}$ denotes the $|A| \times |B|$ submatrix consisting of all entries of L whose row index is in A and whose column index is in B. Thus we have $K(L, V_1) = L_{V_1,V_1} - L_{V_1^c,V_1^c} L^{-1}_{V_1^c,V_1^c} L_{V_1^c,V_1}$. The reduced graph is then specified as $L^{Kron} = K(L, V_1)$, $W^{Kron}_{ij} = -L^{Kron}_{ij}$ if $i \neq j$ and 0 otherwise, and E^{Kron} to be the set of edges with non-zero weights in W^{Kron}. The Kron reduction has a number of desirable properties including: (a) L^{Kron} is indeed a graph Laplacian. (b) If the original graph G is connected, then G^{Kron} is also connected. (c) If G is loopless, then so is G^{Kron}. (d) Two vertices $i, j \in V_1$ are connected by an edge if and only if there is a path between them in G whose vertices all belong to $\{i, j\} \cup V_1^c$. (e) Spectral bounds: $0 = \lambda_{min}(L) = \lambda_{min}(L^{Kron}) \leq \lambda_{max}(L^{Kron}) \leq \lambda_{max}(L)$ (f) For all $i, j \in V_1$, $W^{Kron}_{ij} \geq W_{ij}$ (g) Ring and grid topologies are conserved under Kron reduction.

The third and final step in graph downsampling is sparsification, which is required because the reduced graph ob-

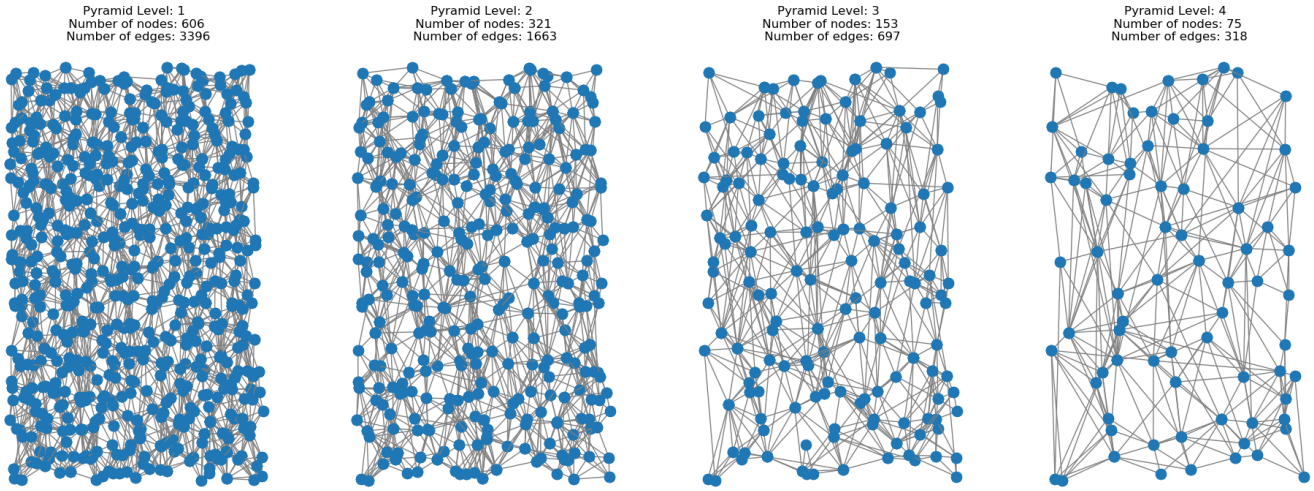


Figure 7: Multiple iterations of downsampling a sensor network graph.

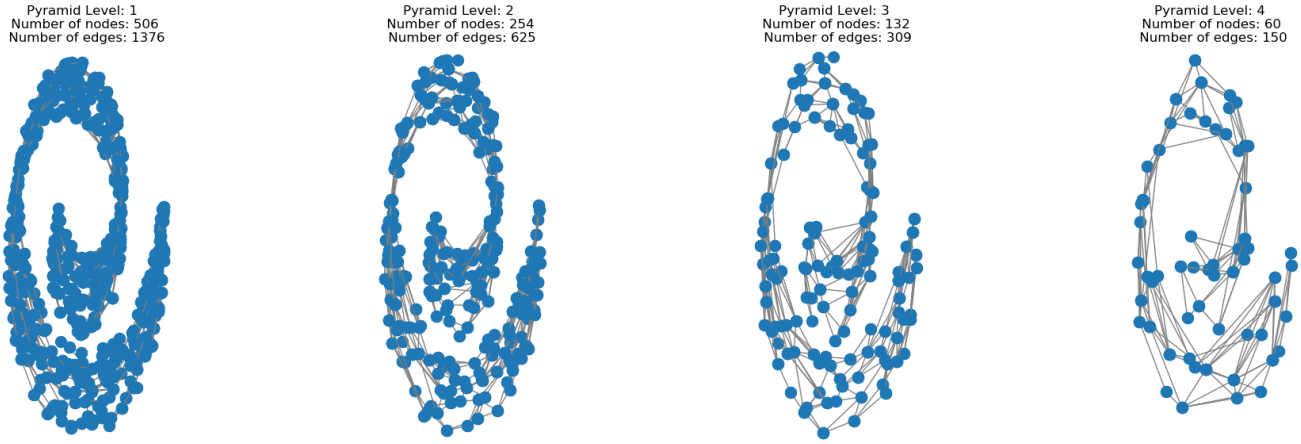


Figure 8: Multiple iterations of downsampling a swissroll graph.

tained after Kron reduction is often very dense. We choose to use the graph sparsification method proposed in [6], which is an efficient $O(n \log n)$ algorithm. It uses the idea of effective resistance from electrical circuits: interpret $G = \{V, E, W\}$ as an electrical network in which edge e has conductance equal to the corresponding edge weight w_e. Then the effective resistance R_e across this edge is the potential difference induced across it when a unit current is injected at one end of e and extracted at the other end of e. A fast method of computing the required effective resistances can be found in Section 4 of [6]. The sparsification algorithm is then given as: Choose a random edge e of G with probability p_e proportional to $w_e R_e$, and add e to the output graph H with weight w_e/qp_e, where q is the total number of edges we wish to retain in H. Take q samples independently with replacement, summing weights if an edge is chosen more than once. The output of this downsampling scheme iterated multiple times to form a pyramid of graphs is visualized in Fig. 7 for the graph of a sensor network, and in Fig. 8 for a swissroll graph.

We note that many other downsampling methods have been proposed in the literature and no one scheme has all the properties that we desire from a downsampling operation. Some examples include: (i) Search for connected subgraphs, replace each such subgraph with one supernode [8]. (ii) Generate a Maximum Spanning Tree using max-cut which naturally defines a multiresolution [3]. (iii) Dilate the GFT spectrum, reduce the graph (say using Kron reduction) and then invert this dilated spectrum onto the reduced graph in an attempt to explicitly enforce the desired spectral properties of the downsampling operation [7]. (iv) Cluster by coarse-graining a Markov Random walk on the graph while attempting to preseve a diffusion operator [2].

2.4. Multiresolution Path Planning

Having described the various operations that we need in the graph domain, we in this section describe how to implement multiresolution path planning for arbitrary graphs. Let the bot be at node n_t at time t. We then convolve the graph signal on the annulus $A(n_t, r_j, r_{j+1})$ with the corresponding scaling function $\phi(j)$. Note that as with the grids in the previous section, we only have to do this once if we are willing to store multiple copies of the map, with each copy corresponding to convolving the graph with the scaling function at one resolution. There would thus be as many maps as there are scales in our multiresolution scheme, forming a pyramid of graphs. We then downsample this annulus j times (for a total reduction in size by a factor of 2^j) using the downsampling scheme described above. Next, we join the reduced graphs for these annular regions together while preserving connections to and from each annulus to generate our final reduced graph. Thus if there is some annulus with nodes p and q that were neighboring nodes in the original annulus graph (they had an edge joining them) but only p was retained in the reduced annulus graph while q was not: we retain the edges to and p from other annular regions and also add the analogous edges of q onto the node p. In this way, the Kron reduction preserves intra-annulus connectivity while we manually ensure inter-annulus connectivity. We can subsequently use Dijkstra's algorithm to perform multiresolution path planning on arbitrary graphs, as desired.

3. Conclusions

We successfully implemented a multiresolution path planning algorithm on grid-like graphs for three choices of scaling function: Haar, Gaussian and Daubechies. The exponential reduction in the number of nodes and edges in the downsampled map afforded a massive speedup to Dijkstra's algorithm. We then proposed a novel multiresolution path planning algorithm for arbitrary graphs using ideas from Graph Signal Processing. Next, we visualized Mexican hat and Meyer wavelets on graphs with interesting topologies. Finally, we demonstrated the efficacy of our graph downsampling scheme that is based on Kron reduction, followed by effective-resistance based sparsification. The simulations show that the resultant downsampled graphs preserve the topology of the original graph while retaining exponentially fewer nodes and edges.

References

[1] R. V. Cowlagi and P. Tsiotras. Multi-resolution path planning: theoretical analysis, efficient implementation, and extensions to dynamic environments. In *49th IEEE Conference on Decision and Control (CDC)*, pages 1384–1390. IEEE, 2010.

[2] S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1393–1403, 2006.

[3] H. Q. Nguyen and M. N. Do. Downsampling of signals on graphs via maximum spanning trees. *IEEE Transactions on Signal Processing*, 63(1):182–191, 2014.

[4] D. I. Shuman, M. J. Faraji, and P. Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2015.

[5] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[6] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[7] Y. Tanaka. Spectral domain sampling of graph signals. *IEEE Transactions on Signal Processing*, 66(14):3752–3767, 2018.

[8] N. Tremblay and P. Borgnat. Subgraph-based filterbanks for graph signals. *IEEE Transactions on Signal Processing*, 64(15):3827–3840, 2016.

[9] P. Tsiotras and E. Bakolas. A hierarchical on-line path planning scheme using wavelets. In *2007 European Control Conference (ECC)*, pages 2806–2812. IEEE, 2007.

[10] P. Tsiotras, D. Jung, and E. Bakolas. Multiresolution hierarchical path-planning for small uavs using wavelet decompositions. *Journal of Intelligent & Robotic Systems*, 66(4):505–522, 2012.

[11] H. Wang, Y. Yu, and Q. Yuan. Application of dijkstra algorithm in robot path-planning. In *2011 second international conference on mechanic automation and control engineering*, pages 1067–1069. IEEE, 2011.