Efficient Multiresolution Scrolling Grid for Stereo Vision-based MAV Obstacle Avoidance

Eric Dexheimer, Joshua G. Mangelson, Sebastian Scherer, and Michael Kaess

***Abstract*— Fast, aerial navigation in cluttered environments requires a suitable map representation for path planning. In this paper, we propose the use of an efficient, structured multiresolution representation that expands the sensor range of dense local grids for memory-constrained platforms. While similar data structures have been proposed, we avoid processing redundant occupancy information and use the organization of the grid to improve efficiency. By layering 3D circular buffers that double in resolution at each level, obstacles near the robot are represented at finer resolutions while coarse spatial information is maintained at greater distances. We also introduce a novel method for efficiently calculating the Euclidean distance transform on the multiresolution grid by leveraging its structure. Lastly, we utilize our proposed framework to demonstrate improved stereo camera-based MAV obstacle avoidance with an optimization-based planner in simulation.**

I. Introduction

As micro-aerial vehicles (MAVs) become more prevalent in the real-world, there is still a need for efficient on-board algorithms. Specifically, navigation in unknown and cluttered environments remains an active area of research. The major components of autonomy, such as localization, mapping, and planning, are all interdependent. Vision-based tracking requires smooth trajectories, low-drift odometry is needed to produce consistent maps, and planning needs an adequate map representation to generate safe and smooth trajectories.

Volumetric representations are frequently used to fuse data temporally into a spatially-organized map. Scrolling local grids maintain a constant memory footprint and can represent distance information densely, which is useful for trajectory optimization. However, the memory consumption of dense grids forces a trade-off between resolution and range for MAVs. Finer resolution is necessary for planning in cluttered environments, and longer-range is needed for fast and smooth flight. Although the limited range of these dense grids is compatible with RGB-D sensors, other sensors, such as passive stereo cameras, provide longer-range distance measurements.

To address this issue, we propose an efficient multiresolution scrolling grid for robot mapping and planning. Detail can be preserved near the MAV and grid limits can be extended while avoiding storage and computational constraints. A naive multiresolution grid implementation would perform independent map updates at each resolution before fusing the levels together. However, this requires computing redundant data and ensuring consistency at level boundaries. We propose an improved alternative which tracks level boundaries during sensor data integration and transfers data between layers as the grid scrolls.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {edexheim, jmangels, basti, kaess}@andrew.cmu.edu

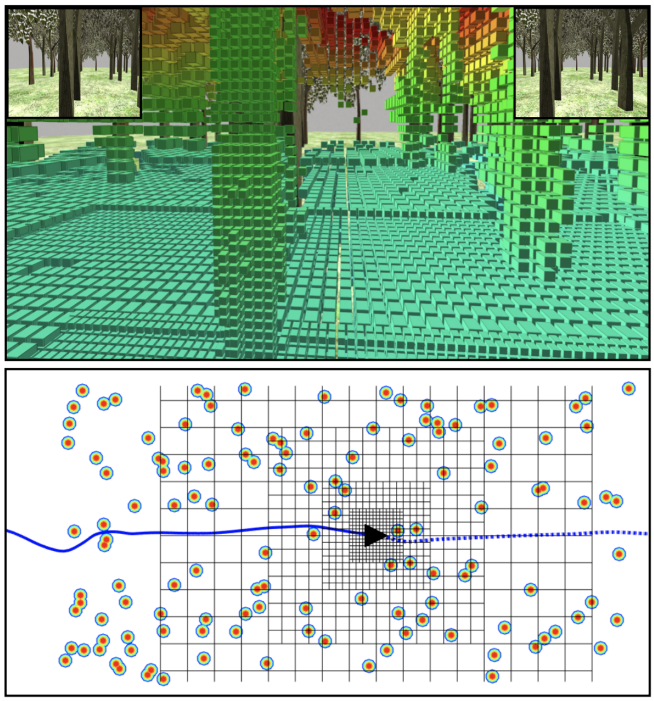

Fig. 1: *(Top)* 3D multiresolution occupancy grid colored by height in simulated environment along with stereo images. *(Bottom)* Top-down view of multiresolution scrolling grid. The goal of the discrete grid is to approximate the true distance field for path planning.

While occupancy is sufficient for finding collision-free paths, the Euclidean distance transform (EDT) can provide gradient information for optimization-based planners to generate smooth paths. For multiple resolutions, a naive EDT computation would update each level before merging the results. Considering all levels jointly is difficult because of irregular dependencies with multiple levels present. We propose a novel method that handles these dependencies and leverages the grid's structure in order to improve efficiency.

Our main contributions are:

- A structured multiresolution occupancy framework that avoids integrating redundant information and handles data transfer between layers.
- A method for efficiently calculating the multiresolution EDT by exploiting the grid's structure.
- Simulation results demonstrating the memory and performance improvements over a single-resolution grid, as well as improved safety and smoothness of planned trajectories.

II. Related Work

A robot's map representation depends heavily on the task at hand. For collision avoidance, free-space information is needed. Obstacles can be represented by points in 3D space, such as those obtained from stereo triangulation. NanoMap performs a greedy uncertainty-aware point obstacle search in a history of frames, which is susceptible to stereo outliers [1]. Obstacle avoidance directly in disparity space has also been proposed [2], and to reduce the influence of noise, a pose graph for a history of disparity frames was introduced in [3]. While polar representations, such as egocylinders [4] and spherical maps [5], can filter out noise with temporal fusion, they are expensive to center around the MAV. Along with single-image methods, they are also subject to occlusion.

In contrast to point-obstacles, occupancy grids store free-space information in a spatially organized structure such that noisy sensor data can be fused temporally [6]. However, uniform grids consume large amounts of memory for large areas or fine resolutions. Better memory efficiency in global maps can be achieved by multiresolution maps, such as octrees, which prune repetitive spatial information [7]. In [8], a local dynamically-tiled octree is constructed, and cached tiles are loaded upon revisiting an area. Hierarchical voxel hashing reconstructs surfaces via a truncated signed distance field (TSDF) using varying resolutions of sparsely allocated blocks [9]. These methods require dynamic memory allocation that can grow without bound, which reduces efficiency and cannot support long-term MAV flight. To maintain constant memory usage, a dense scrolling volume can be represented using a 3D circular buffer [10]. While the local grid is suitable for the limited range of RGB-D sensors, it cannot be extended to the range of stereo cameras without sacrificing significant memory or resolution.

Although occupancy and point-based obstacle information is sufficient for determining if query points are in collision, it does not provide a smooth gradient for continuous-time planning. Trajectory optimization is beneficial for local MAV replanning since it avoids discretization of the state space and permits smooth flight, which in turn reduces failures in state estimation and mapping [11], [12]. Similar to occupancy, the EDT is defined over a spatial grid, but provides the distance to the nearest obstacle [13]. In [12], a batch EDT is obtained from a scrolling occupancy grid to provide uniform B-spline trajectory optimization with a smooth obstacle cost. Incremental distance transforms have also been proposed to update scrolling grids [14] and larger environments on-board MAVs [15]. However, these usually assume static environments, require a priority queue, and may touch the same cell multiple times per update. In contrast, local planning for MAVs should be able to update the representation quickly even with large changes in the environment. Since local trajectory optimization requires dense distance information, it is well-suited for a scrolling map representation.

In [16], a static multiresolution TSDF and EDT is used to map specific areas of a room with varying detail. The GPU processing affords the use of extra computation, such that the TSDF and EDT layers are updated independently before merging the results. Similar to [16], we organize all resolutions into a robocentric grid for efficient mapping. However, our method is for CPU processing, so we do not process redundant information. We also handle independently moving and potentially misaligned layers. A layered 3D circular buffer data structure has been proposed for multiresolution surfel-grids [17], which maintains a history of point measurements and surfels for LiDAR scan registration. Unlike [17], we do not handle layers independently or maintain a history of previous measurements for odometry, and instead focus on constant memory usage. We also require efficient occupancy and EDT updates that avoid processing and storing redundant information.

III. Structured Multiresolution Occupancy Mapping

We construct a multiresolution occupancy map by layering dense 3D occupancy grids, as shown in Fig 1. Starting from an initial resolution for the finest grid at layer $l = 1$, the resolution doubles for each subsequent layer. The grids do not rotate, and independently center themselves around the robot, which triggers data transfer between layers. Since each layer is a dense grid, coarser cells occluded by finer ones exist in memory, but are not maintained to avoid redundancy.

A. Data Structure

Each layer is a 3D circular buffer stored as a 1D array. Additionally, each level has its own offset $\mathbf{o}^l = (o_x^l, o_y^l, o_z^l)^T$ specifying the origin of the grid in the world frame. Circular buffers avoid copying large amounts of data when the map translates, as only the offset and trailing cells leaving the map need to be updated. By keeping the scrolling of layers independent, finer resolutions will scroll more frequently than coarser ones. While scrolling at the coarsest resolution could simplify other operations such as raycasting and the EDT calculation, it would limit the finer resolution mapping area in the robot's field of view.

Similar to [12], we restrict the grid dimensions to be powers of two so that we can use bitwise operations. However, we allow the z-dimension to differ from the xy-dimensions. To index into layer l of the grid, we first obtain a voxel's world coordinate $\mathbf{v}^l = (x^l, y^l, z^l)^T$ by translating the grid coordinates $\mathbf{g}^l$ to the world frame via the offset $\mathbf{o}^l$:

$$\mathbf{v}^l = \mathbf{g}^l + \mathbf{o}^l. \tag{1}$$

For a layer l with grid dimensions $\mathbf{d} = (d_x, d_y, d_z)^T$, and world coordinate $\mathbf{v}^l$, we can obtain the 3D circular buffer index along the x-axis $\bar{x}^l$ as:

$$\bar{x}^l = x^l \ \& \ (d_x - 1). \tag{2}$$

To find the 3D index, we repeat the above for the y^l and z^l coordinates, and then find the 1D array location:

$$i^l = \bar{x}^l d_y d_z + \bar{y}^l d_z + \bar{z}^l. \tag{3}$$

By indexing into the grid using world coordinates, it is also straightforward to index parent and children cells. A parent

cell is a coarser-resolution cell containing the current cell, while a parent cell has eight finer-resolution children cells. The parent cell index n levels above, x^{l+n}, can be obtained by a right arithmetic bit-shift in each dimension

$$x^{l+n} = x^l \gg n. \tag{4}$$

Similarly, the children cells can be found by performing a left bit-shift and iterating over the resulting $2 \times 2 \times 2$ cube. Note that parent or children cells are not guaranteed to be inside the grid, since they are just virtual locations in the world space. We can check whether world x-coordinate x^l resides inside a grid layer l by the following:

$$\text{isInsideX}(x^l) = !((x^l - o_x^l)\ \&(\sim (d_x - 1))). \tag{5}$$

This can be combined with the y and z dimensions to find whether a voxel lies inside a 2D cross-section or 3D volume.

B. Occupancy Updates

Previous structured multiresolution grids update the occupancy at each level independently before fusing them. To avoid processing redundant information and to lessen the computational load, we always update occupancy at the finest possible level that a ray intersects. Coarser levels further from the robot ensure fewer cells need to be updated, but the raycasting needs to accommodate the change in structure. By centering each grid layer around the robot, we only need to check if the next cell along the ray is within the current level before switching to a coarser level. The structure of the grid avoids having to check the level of every cell, since the ray always marches toward coarser resolutions, which improves performance. At the finest voxel resolution, we perform the Bresenham algorithm [18], which increments the world coordinate along the axis of greatest change at each step. This is efficient since it uses integer operations, and only requires an additional check on the validity of the level and bit-shifts for indexing. Although cells at coarser levels may be flagged more than once, a fully multiresolution Bresenham was less precise and slower in our experiments. We update using a standard log-odds formulation, but also restrict each cell to be updated only once per disparity image. This handles inconsistencies caused by shallow viewing angles of large voxels [7]. Coarser cells further from the robot also naturally provide a level of uncertainty in triangulated measurements.

C. Scrolling

For a single-resolution 3D circular buffer, scrolling only requires that trailing cells and the offset be updated. In our multiresolution grid, we present a method for handling the transfer of data between finer and coarser levels. Since we scroll grid layers independently, they may become misaligned. This needs to be handled explicitly to avoid information gaps along edges.

Occupancy grids only represent the probability of an obstacle residing in a cell, so there is no explicit sub-resolution information. Thus, when scrolling from coarse-to-fine, the problem is ill-posed. Rather than copying the probability from the parent to the children cells, we introduce uncertainty

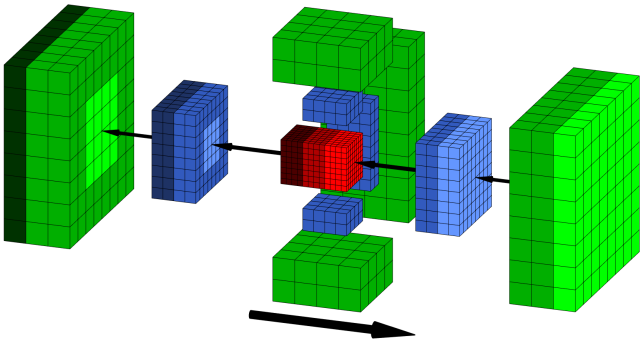

Fig. 2: Example of scrolling between layers with the grid moving to the right. The order of highest resolution to lowest is red, blue, green. Darker colors within a level are cleared, while lighter colors indicate cells being initialized. While not visible, coarser cells that transfer data to finer layers are also cleared.

while maintaining a conservative estimate. Representing the log-odds of a cell at layer l as L^l, and the maximum log-odds L_{max} as unsigned integers, we set

$$L^l = \frac{1}{4}L_{max} + \frac{1}{2}L^{l+1}. \tag{6}$$

This equation ensures that cells retain their occupancy state for a threshold at $L_{max}/2$. However, since the log-odds value is forced closer to the threshold, future measurements will correct the spatial details more quickly. While not theoretically correct, this equation injects uncertainty into the occupancy state without changing it, and also avoids making navigation unsafe. When scrolling from fine-to-coarse, we again use a conservative approach by taking the maximum log-odds from the set of children cells:

$$L^l = \max\{L_c^{l-1} : c = 1 \dots 8\}, \tag{7}$$

where c is the index of the children cell.

The transfer of data between levels must be handled in a specific order to avoid extra copy operations. The following method is summarized in Algorithm 1. First, the offset of the coarsest level is adjusted by finding the scroll difference needed to center the robot in the grid. The cells from the trailing edge in each dimension are cleared, and now represent the leading edges. In the 3D circular buffer, no new memory is allocated, as the shift in the offset only affects the indexing into the 1D array. Next, the trailing edge of the second coarsest level is transferred to the coarsest level using the fine-to-coarse transfer described above and shown in Fig. 2. Since the two levels may not be aligned and partial offsets are possible, all children may not be transferred at the same time. Only the children cells that need to be placed at the leading edge are used. The level offset is updated, and then the coarse-to-fine transfer is performed for the leading edge. All levels including the finest level follow this procedure. Lastly, any coarse cells that become occluded by finer ones are cleared to zero so that the max operation can be correctly applied to cells becoming visible again. This requires the scroll difference from one level lower than that to be cleared.

Algorithm 1 Multiresolution scrolling

Input: Robot position $\mathbf{p}$,
Number of levels L,
Offsets $\mathbf{o}^l, l = 1 \dots L$,
Resolutions $r^l, l = 1 \dots L$
$\mathbf{s}^L \leftarrow$ GetScrollDiff($\mathbf{p}$, $\mathbf{o}^L$, r^L)
$\mathbf{o}^L \leftarrow \mathbf{o}^L + \mathbf{s}^L$
ClearEdges($\mathbf{s}^L$, L)
for l = $L-1$ to 1 **do**
  $\mathbf{s}^l \leftarrow$ GetScrollDiff($\mathbf{p}$, $\mathbf{o}^l$, r^l)
  FineToCoarse($\mathbf{s}^l$, l)
  $\mathbf{o}^l \leftarrow \mathbf{o}^l + \mathbf{s}^l$
  CoarseToFine($\mathbf{s}^l$, l)
end for
for l = 2 to L **do**
  ClearInnerCells($\mathbf{s}^{l-1}$, l)
end for

IV. Structured Multiresolution EDT

While there are previous methods for calculating the EDT on uniform grids, we propose a novel method for efficiently calculating the EDT on a structured multiresolution grid. We avoid performing the EDT on each grid independently, and instead consider all grids jointly, such that redundant information is not processed. Similar to [19], [13], we decompose the calculation into 1D scans along each axis. However, the multiresolution grid has added complexity, since a 1D scan at the coarsest level has dependencies on multiple finer level scans. We first present the batch multiresolution case, where all operations can be done in place on one multiresolution grid. Since the Euclidean and Manhattan distances are equivalent in the first scan, we conduct a two-pass L1 scan in the z-axis to find the distance to the nearest obstacle for each cell along the scan direction. Next, L2 scans in the x-axis and then y-axis build upon previous scans by calculating the lower envelope of parabolas, which yields the 3D EDT. We also discuss speed improvements to the batch case at the expense of additional memory usage.

A. L1 Scan

For the first scan of a binary image, the L1 and L2 distance are equivalent, so we calculate a two-pass L1 distance in the z-axis. The forward pass calculates the distance for free cells that follow obstacles, while the backward pass corrects overestimates in the true distance. This is useful since the L1 scan is significantly faster than L2 and does not require additional memory to store temporary results. Note that we neglect any of the sub-voxel distances in the x and y directions between coarse and fine cell centers.

The forward scan is organized in the xy-plane by cells that overlap with finer levels, and ones that do not. For cells that do not overlap, a forward scan ends after it passes the finer levels. Cells that do overlap are scanned until the next level, and the finer level is initialized. The same strategy is performed through the finest level. At this point, the finer

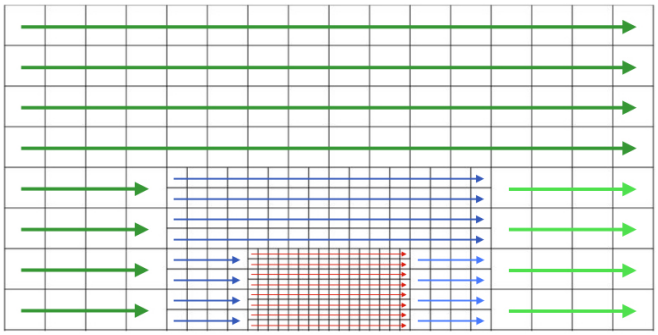

Fig. 3: Forward pass for L1 scan. Darker colors indicate the operation is performed first on that layer, while lighter colors are scans that must be initialized from finer levels. A similar backward pass is also needed to finish the scan.

levels initialize coarser levels with the minimum distance of the children. It can be viewed as a series of branching and merging scans, with a 2D example shown in Fig. 3. The same strategy is used for the backward pass, and all distances are squared to initialize the L2 scans.

When calculating distances across levels, we account for the partial offset between levels. If the partial offset is odd, the edge of the finer level is not aligned with an edge in the coarser level. In the backward scan, finer cells may be an overestimate of the true distance due to the many-to-one dependency between coarse and fine cells in the same scan. This happens when an occupied finer cell propagates distance forward in the scan, which is then "reflected" at the coarsest grid boundary back into adjacent rows. Although this can introduce incorrect distances in the L1 scan, these cells will be corrected in subsequent L2 scans, since they are still overestimates of the true distance.

B. L2 Scans

L2 scans are first conducted in the x-axis, and then in the y-axis. The L2 scans require calculating the lower envelope of parabolas and then filling in the distance values accordingly. To get the lower envelope, the intersection s of two parabolas defined by vertices p and q is needed, which is shown in [13] to be

$$s = \frac{(f(p) + p^2) - (f(q) + q^2)}{2(p - q)}, \tag{8}$$

where f is the squared distance prior to including the current scan. The intersection point s is used to determine whether the current parabola should be added or the previous parabola should be removed from the lower envelope. Additional details of the original algorithm can be found in [13].

In the single resolution case, since there are no levels and the vertices are uniformly spaced, it is straightforward to determine the next vertex and cell index. Thus, the vertex spacing is coupled with the cell index, and can be computed on-the-fly using integer coordinates, with the resolution only needed in post-processing. For the multiresolution grid, the spacing between parabola vertices is not uniform since the resolution varies and there are irregular boundaries when scrolling. To avoid calculating vertex positions, grid indices,

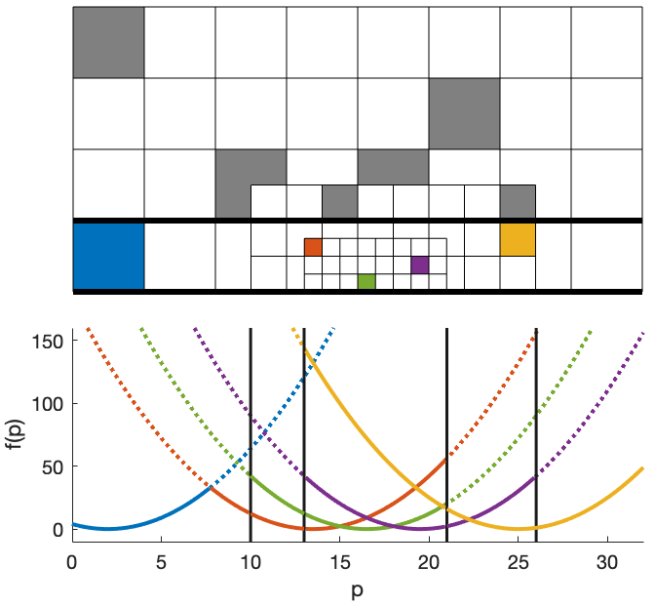

Fig. 4: Multiresolution lower envelope calculation for bottom row of 2D half-grid. Occupancy is shown above, while the the corresponding distance parabolas are below. Lower envelopes are solid lines, while potential parabolas are dotted lines. The vertical lines show boundaries between layers. In the middle, since there are four valid rows in the finest level, each has its own lower envelope, while there are only two in the second level, and one in the coarsest level. Vertices are shown to be zero at obstacle locations, but will take different values when other scans are involved.

and grid levels along the scan direction for all scans, we calculate these values once per EDT update for each direction according to the number of levels a scan must go through. For example, with three levels, the vertices are computed for scans intersecting with one, two, or three levels. Since scrolling will offset the layers with respect to each other, the vertices have to be computed every EDT update. This method saves computation during the lower envelope computation and distance fill-in since the vertices, indices, and levels can be directly accessed from the precomputed vectors.

The L2 scans need the distance values to be stored in a separate array before placing them in the grid, since they would otherwise introduce errors in the parabolic intersections. Thus, to avoid large amounts of temporary memory, the L2 scans are organized by scanning over cells at the coarsest level as shown in a 2D example in Fig. 4. However, we must calculate the lower envelope at the finest cell levels independently, and then merge by taking the minimum distance among scans that overlap at coarser levels.

In the 3D case, the scan cross-section is 2D, so we organize the cell dependencies using 2D Morton decoding as shown in Fig. 5. With a grid containing three levels, the lower envelopes are independent at the first level. In the second and third levels, a maximum of four and sixteen cells, respectively, must be merged by taking the minimum distance. We also need to monitor cases with partially overlapping cells. A check is done to see if a cell at the finest level used is actually inside the grid, and if not, the corresponding parent cell is used when scanning, as in the blue layers in Fig. 5.

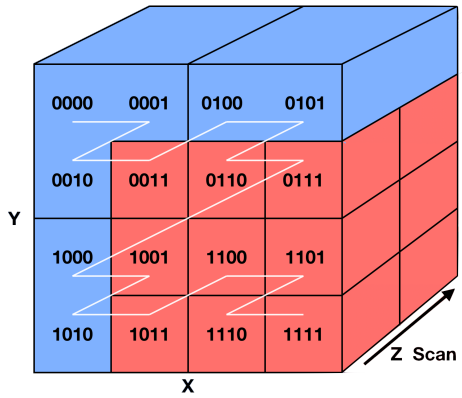

Fig. 5: Organizing L2 scans at the finest possible level in the 3D case. Morton decoding takes the odd bits as the x-coordinate, and the even bits as the y-coordinate. This yields the iteration pattern shown by the white line, which is used to enforce dependencies among scans for coarser levels. In this case, twelve distinct L2 scans must be completed and merged for the coarsest level shown.

This ensures full coverage of the multiresolution grid and avoids calculating the same lower envelope more than once when the layers are not aligned.

C. Beyond Full Batch Solution

We also examine a method for speeding up the distance transform when few cells are updated, without adding significant complexity. For robotic planning, optimization-based planners typically only require a distance up to a maximum value, after which no penalty is incurred in the cost function. Thus, when cells switch from occupied to unoccupied and vice versa, only cells within the truncation distance of these modifications need to be updated. For each of the three axes, a 2D grid at the coarsest cell-level is used to indicate changes. When a cell in a cross-section flips, the indicator grid is dilated by the truncation distance. This ensures all affected distance cells will be updated. The L1 scan is still computed from scratch at each update because it is not organized at the coarsest level and is relatively fast.

Instead of one 3D grid being updated in-place, a multiresolution distance transform grid is needed for each axis to maintain cells that do not need to be updated. Since our distance transform has no knowledge of the nearest obstacle for each cell, scrolling must be handled conservatively. Scrolling across layers from coarse-to-fine is handled by incorporating the squared distance between the parent cell center and the finer cell centers in the cumulative scan directions. Therefore, the distance along the z-axis is used for the first grid, while the complete distance is used for the third grid. Since updating the distance requires a non-constant square root in the former two cases, the scroll-time increases. However, this increase in scrolling time is a very minor performance hit compared to the speedup achieved in the EDT step since fewer cells need to be updated. For fine-to-coarse scrolling, the maximum distance is used, which means that similar to the occupancy case, coarse cells that are occluded must be cleared as well. Lastly, to ensure that the leading edge of the coarsest level is correct, the distance

of newly cleared cells and neighbors within the truncation distance are updated whenever scrolling occurs.

V. Results

We demonstrate the memory and performance properties of the proposed representation compared to a baseline method. Simulated results in a realistic environment with stereo data also demonstrate the use of an expanded grid range with fine resolution near the robot, as the safety and trajectory smoothness are improved.

A. Experimental Setup

A Gazebo simulation environment with textured random forest generation from [11] was used for all experiments. Realistic MAV dynamics and sensors are simulated using RotorS [20]. Noisy image data is also generated, which is then used for stereo matching. This results in realistic noise profiles of the 3D points used for mapping. An example of the environment is shown in Fig. 6.

B. Memory Considerations

We compare the memory required to map a fixed volume with a single-resolution grid and our multiresolution map. For the uniform grid, we have the number of cells m_1:

$$m_1 = d_x d_y d_z. \tag{9}$$

With N layers that double in resolution, the number of cells needed to map the same-sized volume is

$$m_N = N \frac{m_1}{2^{3(N-1)}}. \tag{10}$$

Assuming storage of an occupancy grid (uint8), a flag grid (uint8) for updating occupancy, and the three Euclidean distance grids (float), then the uniform 64^3 grid is 3.670 MB. Naively extending the map range by increasing the dimensions to 256^3 results in 64x memory, while the same space can be mapped using our method with three levels in only 3x memory.

C. Timing

To demonstrate the efficiency of the proposed method, we compare the timing of the framework against [12] in Table I. The timings are gathered from simulation runs in RotorS [20] using the uniform B-spline trajectory optimization from [12]. Increasing the dimension of a single-resolution grid quickly becomes infeasible, as shown by the 256^3 grid. Specifically, the distance transform calculation is not suitable for real-time, and the raycasting is significantly slower as well. For the trajectory optimization, the greater look-ahead properties of the map allows for better initialization and faster convergence in future steps. Even with the increased sensing range, the multiresolution grid provides comparable performance. Note that although the scrolling and raycasting times increase over the single-resolution 64^3 grid, these steps are much less expensive than the EDT update and trajectory optimization. The total times are comparable, even though the mapped area is much larger using the multiresolution grid. The additional memory needed over the single 64^3 grid is not significant, and the overall computation is similar as well.

Step	[12] 64^3	[12] 256^3	Ours 64^3 $L=3$	Ours $64^2 \times 32$ $L=3$
Scrolling	<0.1	1.6	<0.1	<0.1
Occupancy Update	1.0	8.2	3.9	3.4
EDT Update	11.6	631.3	25.8	18.2
Traj. Optimization	33.8	27.8	23.6	20.7
Total	46.4	668.9	53.3	42.3

TABLE I: Average timing comparison in milliseconds between [12] and our method. Raycasting is performed on disparity images downsampled to 160×120. Each grid from [12] uses a 0.15m resolution, and we use the same for the finest resolution. Thirteen control points were used for the trajectory optimization. Total average time assumes mapping and planning run at same frequency.

D. Planning Simulation

To evaluate the multiresolution grid's impact on the safety and smoothness of flight, we conduct simulation experiments in randomly-generated forests. Fifty realistic 50m $\times$ 50m forests with a density of 0.1 trees per meter squared were generated using a modified version of the script from [11]. For the "Baseline" grids, we use 64^3 uniform grids from [12] with 0.15m and 0.30m voxel resolutions. For the multiresolution grid, referred to as "Multigrid" in the results, we choose the dimensions for our method to be $64^2 \times 32$ with 3 levels since the timing best matches that of the uniform grid and it is only a 50% increase in memory. The finest resolution of the multiresolution grid is also set to 0.15m. We use the B-spline trajectory optimization framework from [12], and give a straight-line global trajectory through the forest to guide the local replanning. The max velocity is also varied from 1m/s to 3m/s in increments of 0.5m/s, so that for each max velocity and environment, ten trials are conducted, resulting in 2500 trials per map representation. All occupancy, distance transform, and planning parameters are matched to isolate the representation differences. To measure smoothness, we use the integral of squared jerk over the cumulative planned path [21]. We denote the cost as

$$J = \int_{t_{\text{start}}}^{t_{\text{end}}} \left\| \mathbf{p}^{(3)}(t) \right\|^2 dt. \tag{11}$$

A qualitative example of the uniform and multiresolution grid simulation runs can be seen in Fig. 6. The overall success rates, mean smoothness, and standard deviation of smoothness are written in Table II. Success fractions and smoothness boxplots per max velocity are shown in Fig. 7 and Fig. 8, respectively. Altogether, our method has a higher success rate and improved trajectory smoothness over both single-resolution grids. In addition, our method results in consistently smoother trajectories. Optimization-based planners are susceptible to local minima, so this explains the relatively low success rates, but these results generally agree with that of [12]. In general, using a search-based planner planner to initialize the trajectory [22] or randomly restarting the optimization initialization as in [11] could mitigate this

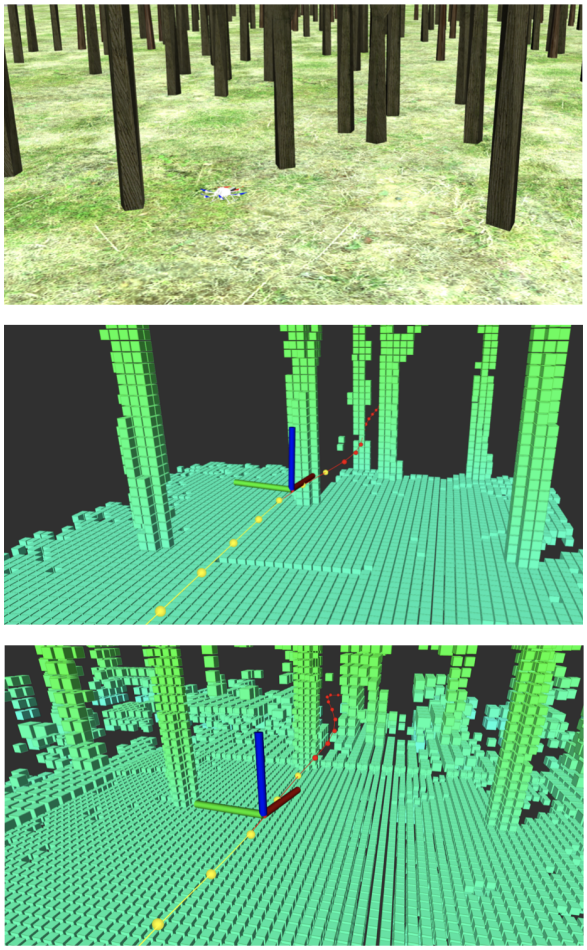

Fig. 6: Simulation views of Gazebo environment using RotorS MAV [20] and planner from [12]. *(Top)* View of realistic simulation forest. *(Center)* Baseline grid plans around first tree, but new trees appear in collision after scrolling, resulting in less smooth trajectories. *(Bottom)* Our multiresolution grid allows for planning further ahead without sacrificing significant memory or computation.

issue. For our purposes, it is sufficient to use only the optimization-based method since it allows for a more direct comparison of the map representations.

	[12] 64^3 0.15 m	[12] 64^3 0.3 m	Ours $64^2 \times 32$ $L = 3$
Success rate	0.538	0.504	**0.636**
Mean J (m^2/s^5)	219.188	196.463	**138.543**
Stdev J (m^2/s^5)	231.421	173.431	**165.410**

TABLE II: Planning simulation statistics comparison over all runs. Mean and standard deviation of the integral of squared jerk are counted only for successful runs.

The smoother trajectories from our method can be attributed to both the increased range over the 0.15m grid, and the finer resolution around the robot compared to the 0.30m grid, which allows for more precise distance and gradient computation in the trajectory optimization. Note that at lower velocities, the finer resolution baseline outperforms the coarser baseline, but as the velocity increases, the coarser baseline performs better due to the further look-

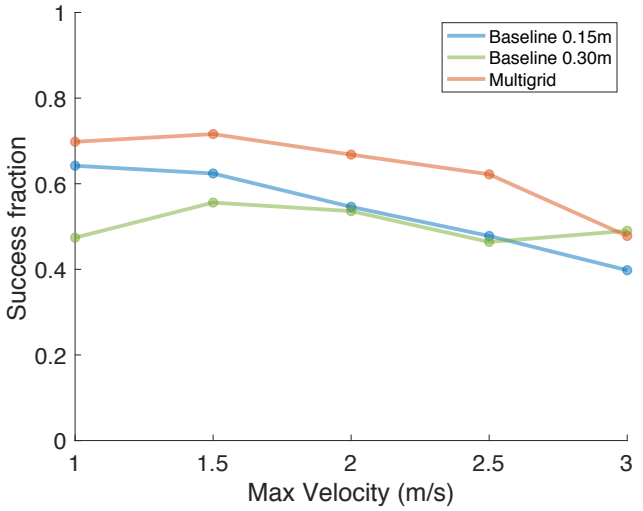

Fig. 7: Success fraction vs. max velocity for different map representations in environments with 0.1 trees per meter squared. Each max velocity contains results across 50 environments and 10 trials per environment.

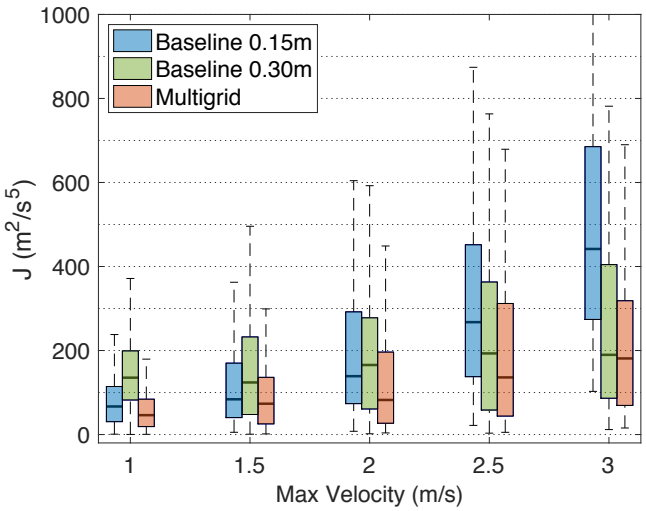

Fig. 8: Box-plots of integral of squared jerk vs. max velocity for different map representations in environments with 0.1 trees per meter squared. Each max velocity contains results across 50 environments and 10 trials per environment.

ahead distance. In terms of smoothness, the multiresolution grid outperforms both baselines across all velocities, and the success rate is higher, except at 3m/s, where it is comparable with the coarser baseline.

Qualitative examples of 25 successful runs per method with a max velocity of 2m/s are shown in Fig. 9. The 0.15m uniform grid is myopic, and thus has very sharp turns at multiple points in the trajectory. While the 0.30m uniform grid reduces many of these, there are still points, such as at $x = -3m$ and $x = 11m$ to $x = 15m$ where there are consistently aggressive turns between trees. Although the multiresolution grid has a couple of aggressive turns, they are much less frequent than the baselines.

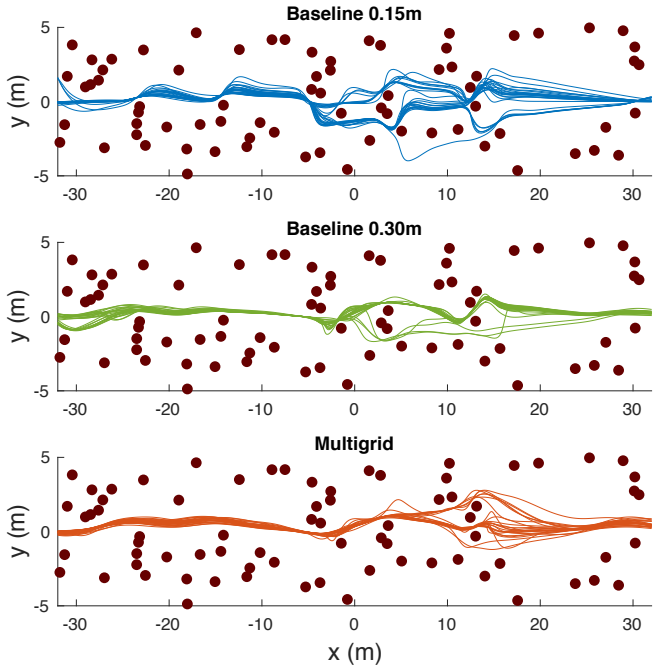

Fig. 9: Qualitative example with max velocity of 2m/s. Each map representation is shown with 25 successful trials. The aspect ratio of the y-axis to the x-axis is 4:1 to enhance visualization.

VI. Conclusion and Future Work

We have investigated the use of an organized multiresolution scrolling grid for occupancy mapping and optimization-based planning. The multiple layers allow for representing the area near the robot at a finer resolution, while also expanding the sensing range. We presented efficient methods for updating occupancy and Euclidean distance information without processing redundant data. Due to its structure, the multiresolution grid provides both computational and memory benefits. Lastly, integration with an optimization-based planner in simulation demonstrated improved MAV flight in unknown, cluttered environments at high speeds.

For future work, using a receding-horizon planner would better suit fast obstacle avoidance, and hardware trials are needed to assess the behavior in different environments. Using the formulation in [23], which defines surfaces based on the occupancy probabilities, would allow for sub-voxel resolution when scrolling between layers. Surface information could also be used for a stereo-based dense odometry. Lastly, incorporating active visual information into the map could be used to supplement the trajectory optimization.

VII. Acknowledgements

The authors would like to thank members of the Robot Perception Lab for their support, especially Paloma Sodhi for insightful discussions. This work was partially supported by DARPA agreement #HR00111820044.

References

[1] P. R. Florence, J. Carter, J. Ware, and R. Tedrake, "NanoMap: Fast, uncertainty-aware proximity queries with lazy search over local 3D data," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 7631–7638.

[2] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 3242 – 3249.

[3] G. Dubey, S. Arora, and S. Scherer, "DROAN — Disparity-space Representation for Obstacle AvoidaNce," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 1324–1330.

[4] C. Cigla, R. Brockers, and L. Matthies, "Image-based visual perception and representation for collision avoidance," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017, pp. 104–112.

[5] P. Gohl, D. Honegger, S. Omari, M. Achtelik, M. Pollefeys, and R. Siegwart, "Omnidirectional visual obstacle detection using embedded FPGA," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 3938 – 3943.

[6] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, 1985, pp. 116–121.

[7] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, 2013.

[8] L. Heng, D. Honegger, G. Hee Lee, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous visual mapping and exploration with a micro aerial vehicle," *J. of Field Robotics*, vol. 31, pp. 654–675, 2014.

[9] O. Kähler, V. Prisacariu, J. Valentin, and D. Murray, "Hierarchical voxel block hashing for efficient integration of depth images," *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, pp. 192–197, 2016.

[10] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul. 2012.

[11] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 5332–5339.

[12] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 215–222.

[13] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, pp. 415–428, 2012.

[14] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, pp. 189–214, 2012.

[15] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373.

[16] R. Wagner, U. Frese, and B. Bäuml, "Real-time dense multi-scale workspace modeling on a humanoid robot," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 5164–5171.

[17] D. Droeschel, J. Stückler, and S. Behnke, "Local multi-resolution representation for 6D motion estimation and mapping with a continuously rotating 3D laser scanner," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 5221–5226.

[18] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, pp. 25–30, 1965.

[19] A. Meijster, J. B. T. M. Roerdink, and W. H. Hesselink, *Mathematical Morphology and its Applications to Image and Signal Processing*, 2002, ch. A General Algorithm for Computing Distance Transforms in Linear Time, pp. 331–340.

[20] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[21] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *The Journal of Neuroscience*, vol. 5, pp. 1688–1703, 1985.

[22] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, pp. 3529–3536, 2019.

[23] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, pp. 1144–1151, 2018.