Efficient 3-D Scene Analysis from Streaming Data

Hanzhang Hu Daniel Munoz J. Andrew Bagnell Martial Hebert

***Abstract*— Rich scene understanding from 3-D point clouds is a challenging task that requires contextual reasoning, which is typically computationally expensive. The task is further complicated when we expect the scene analysis algorithm to also efficiently handle data that is continuously streamed from a sensor on a mobile robot. Hence, we are typically forced to make a choice between 1) using a precise representation of the scene at the cost of speed, or 2) making fast, though inaccurate, approximations at the cost of increased misclassifications. In this work, we demonstrate that we can achieve the best of both worlds by using an efficient and simple representation of the scene in conjunction with recent developments in structured prediction in order to obtain both efficient and state-of-the-art classifications. Furthermore, this efficient scene representation naturally handles streaming data and provides a 300% to 500% speedup over more precise representations.**

I. Introduction

We address the problem of scene understanding from 3-D data (*i.e.*, assigning semantic object categories to small 3-D voxels/points, as shown in Fig. 1) when the data is continuously streamed from a sensor on a moving vehicle. In order to obtain high performance predictions, it has been shown that is necessary to use models that encode the structure/relationships of the predictions [1], [2], [3]. However, in the streaming-data setting, the efficient use of these structured models is a challenging problem due to both theoretical and practical issues. As these algorithms rely on analyzing the entire scene, rather than individual points/voxels, it is unclear how to update the various components of the inference process when 3-D points are being continuously streamed from the sensor. For example, many approaches [4], [5], [6], [7] rely on representing the scene with a segmentation and analyzing the resulting groups/regions/segments instead of points. When data is streaming from the sensor, it is unclear how to efficiently insert new data into an existing segmentation without having to recompute the solution from scratch. Furthermore, structured prediction techniques rely on analyzing the entire scene at once, and it is difficult to efficiently update, rather than recompute, the joint solution with the newly streamed data [8].

In practice, we are often forced to make a compromise in the inference process for the sake of efficient predictions. For example, instead of using a segmentation that obeys object boundaries, we might choose a technique that is less precise but more efficient. Additionally, instead of using expressive contextual models, we might limit ourselves to less expressive models with efficient approximate inference algorithms, or even use a simple classifier. In this work, we demonstrate that we do not need to compromise efficiency for performance, or *vice versa*, and that we can generate state-of-the-art contextual classifications at a high enough rate to handle streamed sensor data on a mobile robot. Specifically, we demonstrate that a simple and efficient, yet imprecise, representation of the scene, when used in conjunction with the region-based scene analysis technique from [4], [9], is able to efficiently predict state-of-the-art classifications.

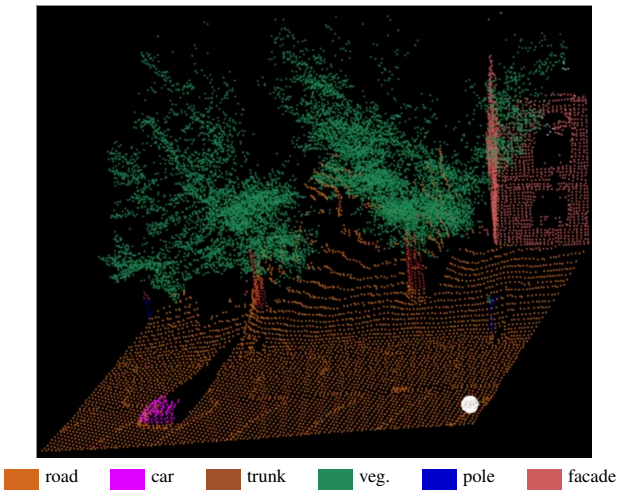


Fig. 1: Screenshot from the supplementary video of classifying streaming 3-D data. The white ball is the sensor location.

The descriptions of our approach are broken down as follows. In Sec. II, we summarize the scene analysis algorithm and its requirements for processing 3-D data. In Sec. III, we describe a data structure that will enable us to efficiently extract regions, perform contextual neighborhood searches, and classify data over large regions of interest around the robot. In Sec. IV, we describe our representations of the scene and how they are used by the scene analysis algorithm. And in Secs. V, VI, and VII, we thoroughly analyze the different aspects of our approach and demonstrate its efficiency and efficacy on real-world datasets.

A. Related Work

In contrast to techniques that perform efficient object detection/classification from streaming 3-D data [10], [11], [12], which often filter out a large portion of the data, we address the problem of efficiently understanding entire 3-D point cloud *scenes*. Related works [13], [14], [15] have

The authors are with The Robotics Institute, Carnegie Mellon University. {hanzhang, dmunoz, dbagnell, hebert}@ri.cmu.edu

This work was conducted through collaborative participation in the Robotics Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016.

similarly focused on scene analysis for robot mobility; however, we address longer range scene understanding, which is important for urban-scale semantic mapping. Additionally, recent works [16], [17] have investigated efficient techniques for classifying streaming point cloud data based on hand-designed filters. The key difference of our work from all the above is that we address the problem of efficient *structured prediction* and can use context in our predictions from a rich set of semantic object categories that would otherwise be difficult to encode using only point cloud descriptors. This work greatly improves upon our earlier work [18] on structured prediction from streaming data. We use a completely different data representation and superior inference algorithm based on our recent work [4].

II. 3-D Scene Analysis via Iterated Predictions

Our approach is based on the iterative inference procedure from [4], [9]. The following description summarizes the method; however, the specific details are not necessary to understand the rest of this paper. 3-D classification is performed using a multi-stage inference procedure over a hierarchical segmentation/representation of the point/voxel cloud. A hierarchical segmentation of the 3-D data consists of multiple segmentations, each of which is referred to as a *level* in the hierarchical segmentation and consists of regions of similar resolution/scale. Levels higher in the hierarchy have coarser segments/regions than levels lower in the hierarchy. The inference procedure operates by traversing up and down levels in the hierarchical segmentation and, for each level, predicting the *distribution* of semantic categories contained within each region. The predictions for each region are influenced by 1) region descriptors that capture the 3-D statistics of the data, and 2) the predictions from the previous level in the hierarchy and spatially neighboring regions, similar to message passing in a graphical model. Predictions from one level are passed to the next and the procedure iterates. The last stage of the inference procedure ends on the finest level of segmentation and the scene is classified by assigning each point/voxel its respective region's object category of highest probability. One important property of this algorithm, which we exploit in this work, is that the technique explicitly models imperfect segmentations and is trained to accurately predict the *distribution* of multiple categories contained within a region.

The two key ingredients of this approach that affect its implementation as an online algorithm are 1) the set of 3-D operations that need to be performed on the point cloud, and 2) the representation of the scene that is fed to the scene analysis algorithm. We stress that the operations and representation are essential and universal to any 3-D scene analysis technique. First, in order to efficiently compute feature descriptors from the point cloud, it is necessary to have a data structure that can perform efficient range search operations over a subvolume in the space. Example standard descriptors, which we also use in our experiments, that require this operation are spin images [19] and local curvature statistics [20]. Second, many techniques use a segmentation algorithm to analyze over 3-D *regions*, instead of individual points [3], [6], [7], [4]. With our inference algorithm, we use multiple (four) segmentations of the point cloud to form a hierarchical segmentation as input. We address these two topics for the streaming data scenario in the following two sections.

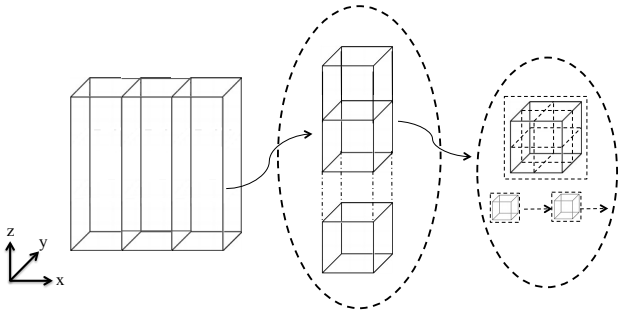


Fig. 2: Visualizations of our data structures. Left: the world is sparsely quantized into infinitely tall *pillars*. Middle: each pillar is sparsely quantized into coarse *blocks*. Right: each block contains a linked list of its occupied *voxels*.

III. Data Structures for Streaming Data

A. Scrolling Grids

One of the prevalent data structures for classifying streaming 3-D data is a scrolling grid representation [15], [13], [14]. Briefly, this data structure quantizes a pre-specified fixed volume of space into a grid composed of n^3 small voxels of prespecified resolution. When robot moves, the indices of the voxels are shifted (scrolled) using a circular buffer. As the size of the grid and resolution of the voxels are known, it is straightforward to insert a point into a voxel inside the grid. Similarly, determining the voxels that constitute a queried subvolume of space can be computed by calculating the min and max extrema voxels and iterating over the subvolume. This data structure is efficient for querying small subvolumes of space; however, when making long-range queries, which is necessary to model contextual interactions among physical objects in the scene, the queried subvolume becomes computationally expensive to iterate over.

B. Sparse Global Grids

Because we need to randomly query very large subvolumes of space, most of which are sparse, we designed a voxel-based data structure to handle this sparsity and still enable efficient long-range query operations. Instead of maintaining a subset of streamed data within a fixed volume, we propose to store all streamed data in a sparse, voxel-based global map. To classify a local map of interest around the robot, we can efficiently query a large subvolume of the space to process with the scene analysis algorithm. Furthermore, as this local map is a subset of the data structure, it still maintains the efficient range search operations necessary for local neighborhood operations needed for feature computation.

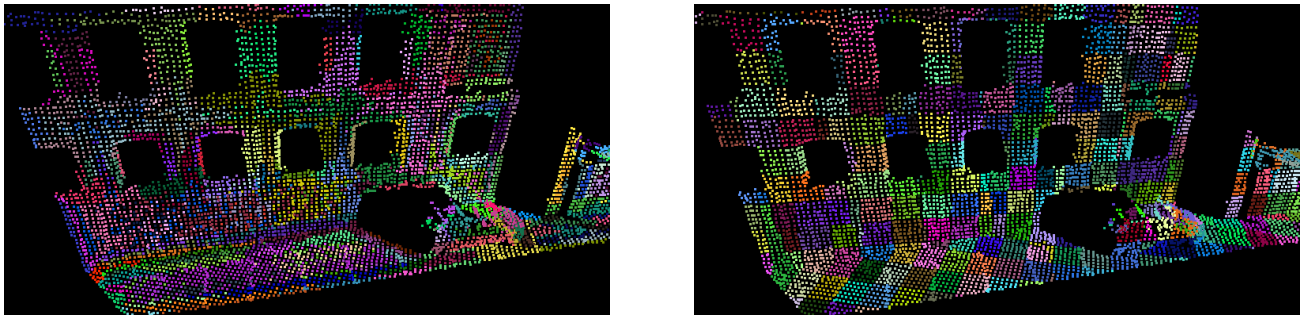


Fig. 3: Example F-H [21] (left) and grid segmentations (right) with a random color per region.

In addition to large subvolume queries, we also require efficient insertion of streamed data over time. Similar to an octree, we consider multiple partitions of the 3-D space to efficiently ignore empty space. Figure 2 illustrates the following explanation. In this work, we use $(0.25\text{ m})^3$ *voxels* as the atomic unit which we assign object categories to. We coarsely partition the 3-D space of voxels into cube-shaped *blocks*, each of which is uniquely indexed based on its global location and stored in a hash map. We can loop over these coarse blocks and retrieve the voxels within to perform queries over large subvolumes, skipping over empty blocks which can reduce search time when the space is sparse. As the block resolution will affect efficiency, we analyze this parameter in Sec. V. We specify the block resolution as an integer multiple of the voxel resolution.

As we want to classify potentially very tall structures in the scene, we want to query local maps of infinite height (z-axis) around the robot. Again, because the space is sparse over large volumes of space, we further group the blocks into *pillars*, each of which is a linked list of non-empty blocks of the same x-y indices and is stored on a hash map. Thus we can efficiently form a local map around a robot by looping over pillars based on an x-y value to retrieve only the non-empty blocks and, thus, the voxels within.

To summarize: each pillar contains a linked list of blocks of the same x-y indices; each block contains a constant-sized 3-D array of voxels and a linked list of non-empty voxels within the block; each voxel contains accumulated statistics of the respective 3-D *points* that fell within the voxel. In our implementation a global map contains a list of non-empty voxels, a hash map of blocks, and a hash map of pillars. Voxels, blocks and pillars are created and updated as needed as new 3-D points are inserted into the global map.

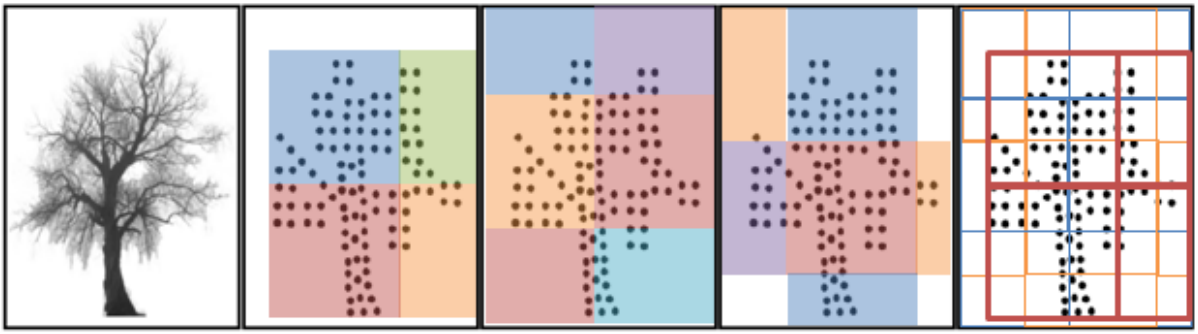
IV. Segmentation

The input to many scene analysis algorithms is a 3-D segmentation. One of the most efficient ways to perform this segmentation (in the batch setting) is the F-H graph-based segmentation algorithm [21]. This algorithm is similar to finding minimum spanning trees and has become prevalent in 3-D applications [22], [23]; an example segmentation is shown in Fig. 3. While F-H is efficient in theory and practice, it does require non-negligible computational costs. First, the algorithm relies on a graph representation, and constructing edges among neighboring nodes in 3-D space relies on local range searches which require non-trivial amounts of time. Second, a notion of similarity between nodes is needed and requires some form of feature computation. Third, in order to incorporate context in predictions, many algorithms [3], [4] rely on using *contextual neighborhoods*, *i.e.*, adjacencies between regions within some fixed radius. As illustrated in Fig. 3, regions resulting from F-H can be irregularly shaped, and accurately computing adjacent regions involves additional range searches. Note that an expanded bounding box approximation would be too crude as regions can be non-convex and/or span extremely large areas of space. Furthermore, filtering points that do lie within some radius of any point within the free-form region may also be costly. Finally, in the streaming data scenario, it is unclear how to efficiently update the previous segmentation with each newly inserted node without having to recompute the segmentation from scratch[1].

Instead of performing a precise segmentation that attempts to obtain object boundaries, we use regions extracted from fixed, gridded partitions of the 3-D space, as also shown in Fig. 3. We refer to this gridded segmentation as a *grid*. This simple approach addresses all of the previous computational concerns: there are no associated setup/construction/feature computations, contextual neighbors can be efficiently found due to all regions having bounded shape, and newly inserted points do not affect the existing segmentation.

When we arbitrarily partition the space, the resulting grid-regions may contain more than one object and/or cut through the boundary of another object. As the per-voxel classification is generated from the finest level segmentation, there will be unrecoverable errors if there exist multiple objects within one region. To address this quantization artifact, similar to how we use a hierarchical segmentation that consists of multiple segmentations at different resolutions in scale, we also consider grids at multiple spatial displacements/offsets from each other. That is, for a grid of fixed resolution, we create multiple grids whose region boundaries are spatially offset from each other in the following way. Suppose an initial grid is constructed with $l \times l \times l$ m^3 resolution regions and that

[1]Although there exists efficient data structures for modifying minimum spanning trees that have complexity sublinear in the number of edges for each online update [24], this would be impractical with streaming 3-D data.



(a) (b) (c) (d) (e)

Fig. 4: (a) An image of the scene. (b) A gridded segmentation of the scene, where each dot represents a voxel and regions are colored. (c,d) Two grids spatially offset from (b). (e) A multi-grid formed by the union of the (b,c,d), where the boundary colors (red, blue, orange) indicate the respective originating grid.

we are considering $\gamma - 1$ different displacements. In the i'th grid, for $i \in \{0, \ldots, \gamma - 1\}$, each of the 3 world coordinates for any region corner has the value $o + \frac{il}{\gamma} + kl$, where $o \in \mathbb{R}$ is the choice of origin and $k \in \mathbb{Z}$ specifies the corner. We refer to the union of the regions from each grid as a *multi-grid*. Although a multi-grid is not a proper "segmentation" due to elements (voxels) being contained within multiple regions, we refer to one multi-grid as a single segmentation in that it is a set of (overlapping) regions. Finally, since a voxel may be contained within multiple regions across displaced grids, the voxel's final label distribution is the unweighted average over all the label distributions of the respective regions it falls into. Figure 4. illustrates a multi-grid with $\gamma = 3$.

V. Efficiency Analysis

In the remaining sections, we compare various performance metrics with using F-H segmentation vs. simple (multi-)grids. Our analysis is performed on the 3-D point cloud datasets `VMR-Oakland` [4] and `Freiburg` [25]. Examples of classified scenes from each dataset are shown in Fig. 5. For the computation analysis in this section, we use the training and validation folds from the `VMR-Oakland` dataset. For classification analysis, we evaluate *voxel* classification error and assign the ground truth label to each voxel as the mode ground truth label from its respective points. All timing results were obtained on an Intel i7-2960XM processor using a maximum of 8 threads.

A. Setup

For each segmentation algorithm, F-H and (multi-)grids, we construct a 4-level hierarchy, from fine to coarse, by varying parameters that affect scale. When constructing the graph for F-H, we use a spatial neighborhood of 0.5 m radius to create edges between two voxels, and we use the Euclidean distance between two feature vectors that encode local geometry and orientation [20]. The specifics of the grid partitions are discussed in the following subsections.

We compute the same four types of region features over the two different hierarchical segmentations. 1) A bag-of-words representation through feature quantization using soft k-means assignment [26] over two per-voxel descriptors: a) 5×5 spin images of $(0.2\text{ m})^2$ cell resolution, b) local geometry and orientation features computed over three local neighborhoods of radii 0.5 m, 0.8 m, and 1.1 m, respectively. 2) Relative elevations using a 2.5-D elevation map. 3) The shape of the region through spectral analysis of the voxel coordinates that constitute the region, weighted by the number of points that fell into the voxel [4]. 4) The region's bounding box statistics [4].

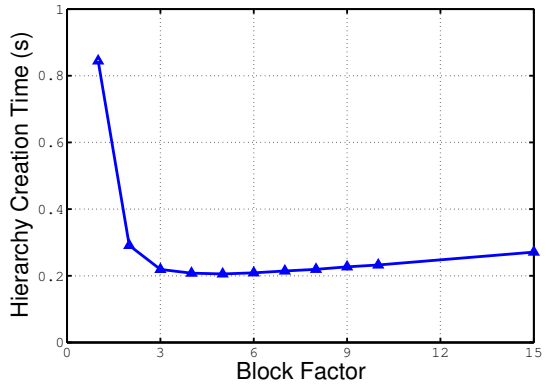


Fig. 6: Average region hierarchy construction time, on validation data, with respect to block factor $(\frac{\text{block-resolution}}{\text{voxel-resolution}})$.

B. Block Resolution

Our global grid uses $(0.25\text{ m})^3$ voxels as the atomic element for classification. As previously mentioned, we perform efficient range searches using coarse neighboring blocks to skip over empty volumes of space. We select the resolution of the blocks by analyzing the construction time of our region grid hierarchy, which is a function of the range searches needed to compute the feature descriptors and contextual neighborhoods. In Fig. 6 we plot the average computation time with respect to coarsening block resolution, which is quantified by the ratio $\frac{\text{block-resolution}}{\text{voxel-resolution}}$ and is referred to as a "block factor". As expected, we observe a block factor of 1, meaning iterating over every neighboring voxel, is

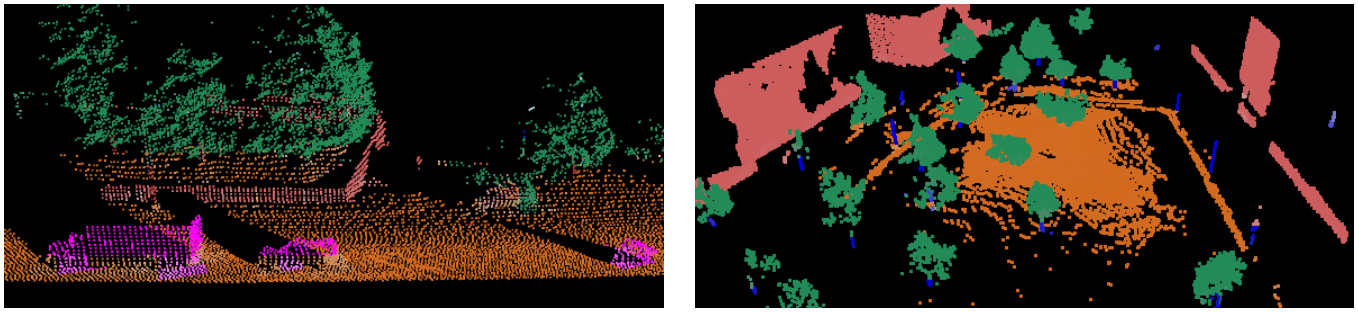


Fig. 5: Example classifications from the `VMR-Oakland` [4] (left) and `Freiburg` [25] (right) datasets.

the slowest. In contrast, we see computation time start to increase when the block resolution coarsens to a factor more than 5. Hence, we use a block factor of 5 in the remaining experiments.

C. Finest Segmentation Resolution

The final voxel classifications are generated from the finest level in the hierarchical segmentation. Ideally, we would choose the finest segmentation so that each voxel is a unique region in order to avoid any quantization artifacts; however, this precision comes at the cost of more samples to classify and increases inference time. On the other hand, using larger regions runs the risk of grouping together different labels within one region. To quantify this mixture of labels within a region, we can compute the average ground truth label entropies for regions with different sizes (from the training set). This value directly relates to the misclassification rate for assigning a single label to a region containing a mixture of labels. In Fig. 7 we plot the trade-offs of entropies (a) and number of generated regions (b) for different region grid sizes. We observe an exponential drop in the number of regions as the cell size increases, which is good for efficiency, and an increasing entropy, which hurts performance. As a compromise to balance efficiency and accuracy, we choose $(1.5\text{ m})^3$ as the grid region resolution in our finest level segmentation. The resolutions of the three coarser segmentations in the hierarchy are less sensitive as it is only the finest level segmentation that assigns the per-voxel labels. We use increasingly coarse regions of $(3.5\text{ m})^3$, $(7.5\text{ m})^3$, $(10\text{ m})^3$ resolution, respectively, for the remaining levels.

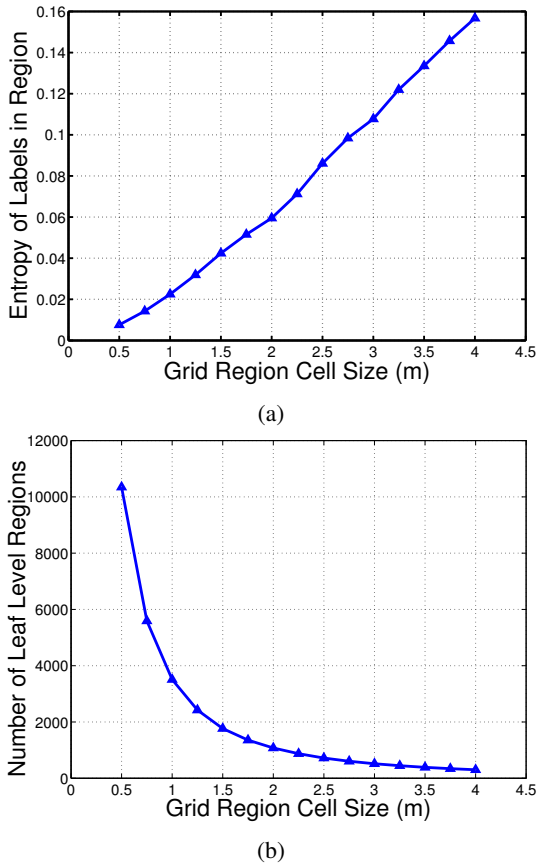


Fig. 7: Analysis, on validation data, of region grid resolution at the finest level. (a) Ground truth label entropy vs. region size. (b) Number of regions vs. region size.

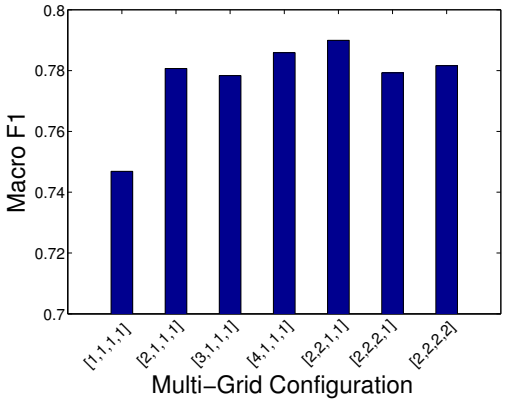
D. Multi-grid Configuration

A multi-grid is the union of multiple grids of the same resolution with spatial displacement from each other. The more grids we have, the more robust we are to arbitrary quantization artifacts. However, this improvement in precision comes at the cost of having more regions in the scene to analyze. In Fig. 8, we analyze, on the validation fold, the behavior of using multi-grids (with various sizes) at different levels in the region hierarchy. As our hierarchy contains four levels, we specify the multi-grid configuration of each level with a 4-tuple $[\gamma_1, \gamma_2, \gamma_3, \gamma_4]$, where γ_ℓ is the number of grids in the multi-grid of level ℓ in the hierarchy and $\ell = 1$ is the finest segmentation level. From Fig. 8-a, we observe a large improvement in performance when simply using 2 grids in the leaf level ($[2,1,1,1]$) vs. using only one ($[1,1,1,1]$) multi-grid. Fig. 8-b shows that this improvement in performance comes at an extra cost of 0.14 s when classifying a scene, on average. However, we also observe diminishing returns in average/macro per-class F_1 performance as we increase the number of grids with respect to computation time. Therefore, we use the $[2,1,1,1]$ multi-grid configuration in the remaining experiments.

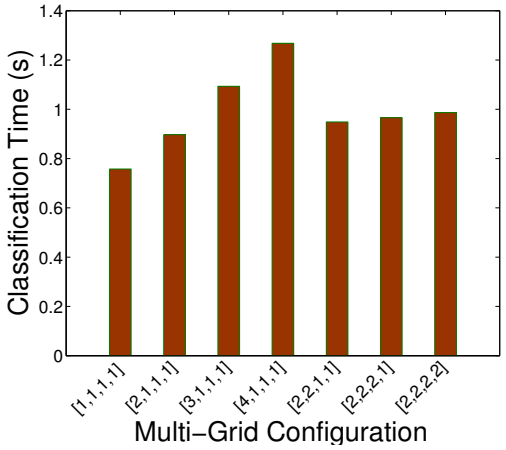
The most costly computation in constructing the region

TABLE I: Decomposition of computations for constructing the hierarchical regions for a Grid and $[2,1,1,1]$ Multi-grid.

	# regions	segmentation (s)	feature (s)	neighbor-context (s)	total (s)	neighbor-context %
Grid level 0	1862.6	0.0100	0.0138	0.0541	0.0779	69.45%
Grid level 1	426.7	0.0100	0.0100	0.0261	0.0461	56.62%
Grid level 2	107.3	0.0082	0.0088	0.0123	0.0293	41.98%
Grid level 3	63.6	0.0064	0.0084	0.0126	0.0274	45.99%
total	2460.2	0.0346	0.0410	0.1051	0.1807	58.16%
Multi-grid level 0	3748.3	0.0197	0.0277	0.0692	0.1166	59.35%
Multi-grid level 1	426.7	0.0105	0.0103	0.0264	0.0472	55.93%
Multi-grid level 2	107.3	0.0074	0.009	0.0125	0.0289	43.25%
Multi-grid level 3	63.6	0.0066	0.0085	0.0128	0.0279	45.88%
total	4345.9	0.0442	0.0555	0.1209	0.2206	54.81%



(a)



(b)

Fig. 8: Analysis, on validation data, of (a) classification performance and (b) computation time with respect to different multi-grid configurations.

hierarchy is determining the contextual neighborhoods for each region. Typically, we perform very large range searches, up to 10 m, in order to model long-range interactions. If we consider the number of grids in a multi-grid, γ, the overall computation time for determining neighboring context over all regions is roughly raised by the number of additional offset grids. To avoid this extra cost, when computing contextual neighborhoods in multi-grids on the regions that are offset by a relatively small distance, we perform the following approximation. We know that regions that overlap have some fixed, equally spaced offset from each other, and that the offsets are small with respect to the context range. Therefore, for regions that overlap each other, the contextual neighborhoods cover essentially the same 3-D space. Hence, instead of computing multiple neighborhoods for every offset region, we compute one contextual neighborhood and share it with its overlapping regions. In Table I, we decompose the average timings for constructing a 4-level region hierarchy using a grid and a $[2,1,1,1]$ multi-grid segmentation. We demonstrate that the use of the neighborhood approximation achieves comparable timing as with using a single grid.

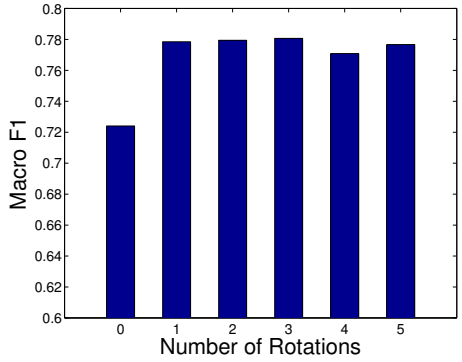
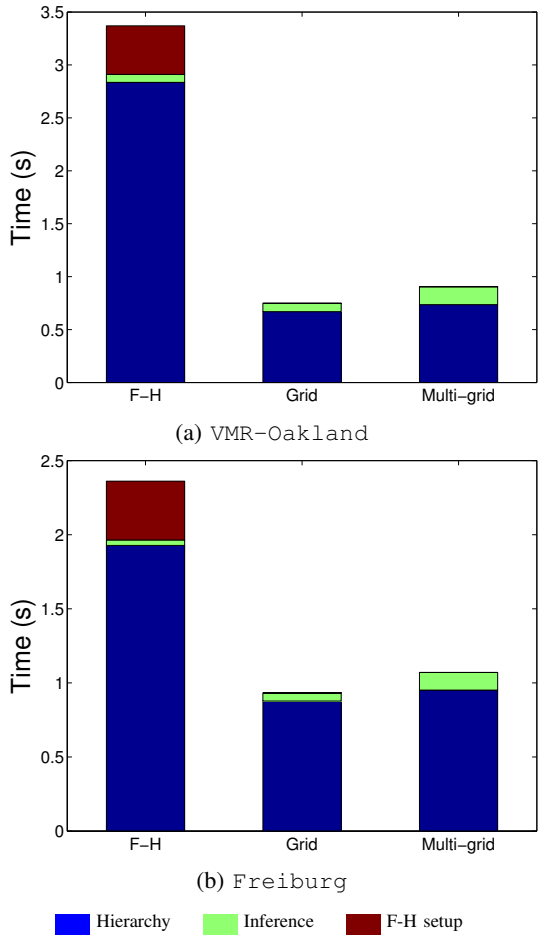


Fig. 9: Average per-class F_1, on validation data, with respect to the number of times the training data is rotated.

VI. Classification Analysis

A. Addressing Quantization Artifacts

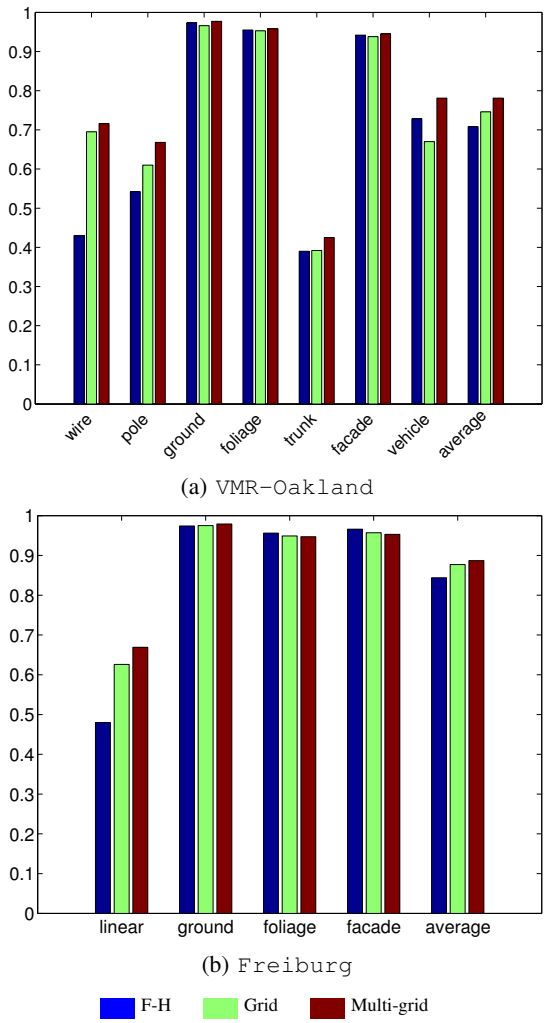
Because our grid representation uses a fixed partitioning of the space, the segmentation is not invariant under rotations/translations of the scene. Note that although precise segmentation algorithms, such as F-H, are invariant to these transformations, the quantized voxels may not be if they are too coarse. We address this problem when training the models for both the F-H and (multi-)grid hierarchies. For each training point cloud, we generate additional training scenes by rotating the original scene around the z-axis with equally spaced angles between $[0,\pi/2]$. In Fig. 9, we quantify on the `VMR-Oakland` validation set the performance with respect to the number of times we rotate each training scene; we use 3 rotations in the remaining experiments.



(a) VMR-Oakland

(b) Freiburg

Fig. 10: Average timings of each component during the entire inference procedure. Hierarchy construction includes feature computation time.



(a) VMR-Oakland

(b) Freiburg

Fig. 11: Per-class F_1 for the datasets. "average" is the mean over the classes.

B. Classification Performance

We now evaluate, on the evaluation folds from each dataset, the efficiency and classification performances of the 3 models trained on different hierarchical region representations: 1) F-H, 2) Grid, and 3) $[2,1,1,1]$ Multi-grid.

In Fig. 10, we break down the computation for each model on the two different datasets. We observe that the grid-based model timings are inversely related with the F-H model. That is, the grid-based region constructions are much faster than F-H; however, F-H compresses the scene into a smaller number of regions which results in a faster inference time. Overall, we observe the average computation time with a multi-grid is 2.5-3x faster than using the more precise F-H segmentation, per static scene.

In Fig. 11, we present the classification rates for the different models, in terms of F_1 scores for each class. We observe that using a grid-based segmentation can exceed the performance of using a precise F-H segmentation. This follows from the property that the scene analysis algorithm [9] is robust to imperfect segmentations due to explicitly modeling the distributions of labels within regions. Additionally, we can further improve performance by using a multi-grid to account for discretization artifacts from a single grid. In conjunction with the previous timing information, we conclude that this is an efficient and effective approach to perform full 3-D scene analysis.

VII. Streaming Classification

We demonstrate the practical benefits of using our efficient representation in the streaming data scenario. Both datasets contain sensor logs collected while the robot was moving. The VMR-Oakland log was collected from a push-broom laser scanner mounted on the side of a moving vehicle, and the sequence is broken down into three smaller logs. The Freiburg log was collected on a wheeled robot moving in a campus environment. The sensor was a pan-tilt laser that scans a 360° field of view, and data was collected in a stop-and-go manner from 77 different scanning locations.

We process each log in the same manner: after inserting the last $10,000$ streamed 3-D *points* into the global voxel grid, we construct a local map of 20 m L_∞ radius, in the x-y plane, centered at the mean coordinate of the newly inserted $10,000$ points. Due to the profile scanning pattern

TABLE II: Video sequence statistics

	VMR-Oakland	Freiburg
Avg. Number of 3-D Points / Scene	44,198	452,330
Avg. Number of Voxels / Scene	10,904	34,031
Total Number of Classified Scenes	398	1,059
Total Distance Traveled (m)	2,950	723

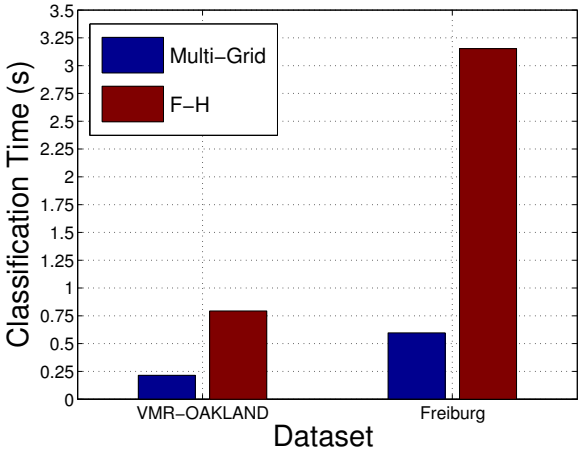


Fig. 12: Average classification time per scene using multi-grid and F-H segmentation on streams of VMR-Oakland and Freiburg datasets.

in the VMR-Oakland dataset, the resulting local map is approximately 20×20 m^2; however, it is a full 40×40 m^2 in the Freiburg dataset. We refer to this local map as a "scene" and then construct the region hierarchy and classify the scene with our scene analysis algorithm.

In Table II, we break down the average number of 3-D points and voxels contained within each scene for each dataset, as well as the total number of classifications needed to process each sequence and how far the robot traveled. The supplementary video shows the processing of each log using the multi-grid model. The video is screen captured in real-time and demonstrates the ability of our approach to efficiently process data for use on board mobile robots. In Fig. 12, we compare the average classification time, per scene, when using a simple multi-grid representation vs. F-H segmentation. Using the F-H representation is 3-5x more expensive than using the efficient multi-grid representation and would greatly limit a robot's speed in practice.

VIII. Conclusion

In this work, we described a simple approach for performing 3-D scene analysis from streaming data. We demonstrated that we do not need to make a compromise in classification performance for the sake of efficiency and can achieve the best of both worlds in practice. Specifically, we showed that we do not need to rely on precise (and computationally more expensive) representations of the scene and can instead use a simple and efficient representation to achieve state-of-the-art classifications.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3d range data," in *CVPR*, 2005.

[2] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *CVPR*, 2009.

[3] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *NIPS*, 2011.

[4] X. Xiong, D. Munoz, J. A. Bagnell, and M. Hebert, "3-d scene analysis via sequenced predictions over points and regions," in *ICRA*, 2011.

[5] B. Douillard, J. P. Underwood, N. Kuntz, V. Vlaskine, A. J. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3d lidar point clouds," in *ICRA*, 2011.

[6] K. Lai and D. Fox, "Object recognition in 3d point clouds using web data and domain adaptation," *IJRR*, vol. 29, no. 8, 2010.

[7] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3D point clouds in urban environments," in *ICCV*, 2009.

[8] P. Kohli and P. H. Torr, "Dynamic graph cuts for efficient inference in markov random fields," *T-PAMI*, vol. 29, no. 12, 2007.

[9] D. Munoz, J. A. Bagnell, and M. Hebert, "Stacked hierarchical labeling," in *ECCV*, 2010.

[10] A. Teichman and S. Thrun, "Tracking-based semi-supervised learning," *IJRR*, vol. 31, no. 7, 2012.

[11] C. Mertz, L. E. Navarro-Serment, D. Duggins, J. Gowdy, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppe, C. Urmson, N. Vandapel, M. Hebert, and C. Thorpe, "Moving object detection with laser scanners," *JFR*, 2012.

[12] M. Himmelsbach, T. Luettel, and H.-J. Wuensche, "Real-time object classification in 3d point clouds using point feature histograms," in *IROS*, 2009.

[13] J.-F. Lalonde, N. Vandapel, and M. Hebert, "Data structures for efficient dynamic processing in 3-d," *IJRR*, vol. 26, no. 8, 2007.

[14] M. Bansal, B. Matei, B. Southall, J. Eledath, and H. Sawhney, "A lidar streaming architecture for mobile robotics with application to 3d structure characterization," in *ICRA*, 2011.

[15] C. Wellington and A. Stentz, "Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetation," in *FSR*, 2003.

[16] O. Hadjiliadis and I. Stamos, "Sequential classification in point clouds of urban scenes," in *3DIMPVT*, 2010.

[17] I. Stamos, O. Hadjiliadis, H. Zhang, and T. Flynn, "Online algorithms for classification of urban objects in 3d point clouds," in *3DIMPVT*, 2012.

[18] D. Munoz, N. Vandapel, and M. Hebert, "Onboard contextual classification of 3-d point clouds with learned high-order markov random fields," in *ICRA*, 2009.

[19] A. E. Johnson and M. Hebert, "Using spin-images for efficient multiple model recognition in cluttered 3-D scenes," *T-PAMI*, vol. 21, no. 5, 1999.

[20] G. Medioni, M. Lee, and C. K. Tang, *A Computational Framework for Segmentation and Grouping*. Elsevier, 2000.

[21] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, 2004.

[22] J. Strom, A. Richardson, and E. Olson, "Graph-based segmentation of colored 3d laser point clouds," in *IROS*, 2010.

[23] R. Triebel, J. Shin, and R. Siegwart, "Segmentation and unsupervised part-based discovery of repetitive objects," in *RSS*, 2010.

[24] G. N. Frederickson, "Data structures for on-line updating of minimum spanning trees," in *STOC*, 1983.

[25] J. Behley, V. Steinhage, and A. Cremers, "Performance of histogram descriptors for the classification of 3d laser range data in urban environments," in *ICRA*, 2012.

[26] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011.