Efficient 3-D Scene Analysis from Streaming Data

Hanzhang Hu  
*Carnegie Mellon University*

Daniel Munoz  
*Carnegie Mellon University*

J. Andrew Bagnell  
*Carnegie Mellon University*

Martial Hebert  
*Carnegie Mellon University*

Follow this and additional works at: [http://repository.cmu.edu/robotics](http://repository.cmu.edu/robotics)

Part of the [Robotics Commons](http://repository.cmu.edu/robotics)
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. We are 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 that also obtains state-of-the-art classification performance. Furthermore, as our efficient representation is naturally suited for streaming data, we demonstrate that we can achieve these high-performance predictions at a rate 3x faster than when using precise representations and without any loss in performance.

I. INTRODUCTION

We address the problem of scene understanding from 3-D point clouds (i.e., assigning semantic categories to each 3-D point, as shown in Fig. 1) when the data is continuously streamed from a sensor onboard of 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]–[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. For example, many approaches [4]–[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 previous 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, can generate state-of-the-art contextual classifications at a high enough rate to handle streamed sensor data from a mobile robot. Specifically, we demonstrate that a simple and efficient representation of the scene, when used in conjunction with the region-based, scene analysis technique from [4], [9], is able achieve state-of-the-art results, even compared to computationally more expensive precise representations.

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 [11]–[13], 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 [14]–[16] have 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 [17], [18] have investigated efficient techniques for classifying streaming point cloud data based on carefully hand-drawn 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 only using point cloud descriptors. This work greatly improves upon our earlier work [19] 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

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. Classification over a point cloud is performed using a multi-stage inference procedure over a hierarchical segmentation of the point cloud. 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 computed over the point cloud, and 2) the predictions from the previous scale 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 scene is classified with the predictions at the finest level of segmentation: each voxel is assigned its respective region’s label 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 labels 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 [20] and local curvature statistics [21]. Second, many techniques use a segmentation algorithm to analyze over regions of 3-D points, instead of individual 3-D points [3], [4], [6], [7]. With our inference algorithm, we use multiple (four) segmentations of the point cloud as input. We address these two topics for the streaming data scenario in the following two sections.

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 [14]–[16]. Briefly, this data structure quantizes a pre-specified fixed volume of space into a grid composed of \( n^3 \) small voxels of pre-specified resolution; in this work, we use 25 cm voxels. 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. As the robot moves, the indices of the voxels are shifted (scrolled) using a circular buffer. This data structure is efficient for querying small subvolumes of space; however, we need to make long range queries in order to properly model contextual interactions in the scene.

B. SPARSE GLOBAL GRIDS

Because we need to randomly query 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 an area of interest, 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.

Figure 2 illustrates these ideas. As we want to classify potentially very tall structures in the scene, we want to query local maps of infinite height (z-axis). Hence we quantize the x-y plane into a 2.5-D grid, where each cell in this grid is another data structure, which we refer to as a pillar, that contains voxels. To avoid the problem of iterating over many (empty) voxels during a large range search, we sparsely quantize the infinitely-tall pillar along the z-axis into smaller volumes called blocks, and a block contains a linked list of the occupied voxels inside it. Therefore, when performing range search over a large volume of space, we loop over non-empty voxels contained in non-empty blocks. As the block
resolution will affect efficiency, we analyze this parameter in Sec. V.

In practice, each block has a unique index with respect to some arbitrary origin. For a queried \((x,y,z)\) coordinate, we can compute its respective block index by first computing its voxel coordinates \((i_j, j_v, k_v)\) by dividing each point coordinate by the voxel resolution, and then dividing the voxel coordinates by the block resolution to obtain block coordinates \((i_b, j_b, k_b)\) to form a unique block index. Blocks are stored in memory using a hash table, and large queried subvolumes can be efficiently retrieved by looping from the min to max block coordinates. Because only a small percentage of voxels are occupied in outdoor scenes, efficiency is improved due to inter- and intra-block sparsity instead of densely looping over each possible voxel.

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 [22] graph-based segmentation algorithm, which is similar to finding minimum spanning trees. This algorithm has become prevalent for 3-D applications [23], [24]; an example segmentation is shown in Fig. 3. While efficient in practice, there are some computational costs to consider. 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 space, as also shown in Fig. 3. 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.

Although there exists efficient data structures for modifying minimum spanning trees that have complexity sublinear in the number of edges for each online update [25], this would be impractical with streaming data.

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 [27]. For the computation analysis in this section, we use the training and validation folds from the VMR-Oakland dataset. Examples of classified scenes from each dataset are shown in Fig. 5. 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 to create edges between two voxels, and we use the Euclidean distance between two feature vectors that encode local geometry and orientation [21]. 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 [28] over two per-voxel descriptors: a) \(5 \times 5\) spin images of \((0.2\) m\)\(^2\) cell resolution, b) local geometry and orientation features computed over three local neighborhoods of radii 0.5, 0.8, and 1.1 m, respectively. 2) Relative elevations using a 2.5-D elevation map. 3) The shape

\(^1\)Although there exists efficient data structures for modifying minimum spanning trees that have complexity sublinear in the number of edges for each online update [25], this would be impractical with streaming data.

\(^2\)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.
Fig. 4. From left to right: (a) An image of an object. (b,c,d) Three independent region grids created from different spatial offsets. (e) A multi-grid formed by the union of the (b), (c), (d), where the boundary colors red, blue, orange respectively indicate the originating grid. The voxel’s label distribution is the unweighted average of all the label distributions of the regions it falls into.

Fig. 5. Example classifications from the VMR-Oakland [4] (left) and Freiburg [27] (right) datasets.

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

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].

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 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 decreasing 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 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 \((3.5, 7.5, 10 \text{ m})^3\) resolution, respectively, for the remaining levels.

**D. Multi-grid Offsets**

A multi-grid is the union of multiple grids of the same resolution with various spatial offsets. The more offsets 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 the behavior of using multiple offsets for different multi-grids in the region hierarchy on the validation fold from the VMR-Oakland dataset. As our hierarchy contains four levels, we specify the number of offsets per level with a 4-tuple \([o_1, o_2, o_3, o_4]\), where \(o_\ell\) is the number of offsets for level \(\ell\) in the hierarchy and level 1 is the finest segmentation level. From Fig. 8-a, we observe a large improvement in performance when simply using 2 offsets 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 offsets 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 hierarchy is determining the contextual neighborhoods for each region. Typically, we perform very large range searches, up to 10 m, to model long-range interactions. If we consider multiple offsets in the multi-grid, the computation time for determining neighboring context is roughly raised by the number of additional offsets. Instead, to speed up the time to compute contextual neighborhoods using the multi-grid, we approximate this contextual neighborhood in the following way. 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.

**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 data; we use 4 rotations in the remaining experiments.

**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.

In Table II, we present the classification rates for the different models. 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 \( L_{\text{max}} \) radius, in the x-y plane, centered at the mean coordinate of the newly inserted 10,000 points. Due to the profile scanning pattern in the VMR-Oakland dataset, the resulting local map is approximately \( 20 \times 20 \text{ m}^2 \); however, it is a full \( 40 \times 40 \text{ m}^2 \) in the Freiburg dataset. We refer to this local map as a
Fig. 11. Average classification time per scene using multi-grid and F-H segmentation on streams of VMR-Oakland and Freiburg datasets.

€“scene” and then construct the region hierarchy and classify the scene with our scene analysis algorithm.

In Table III, 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. Videos showing the processing of each data log using the multi-grid model are available at [10], [29]–[31]. These videos are screen captured in real-time and demonstrate the ability of our approach to efficiently process data for use on board mobile robots. In Fig. 11, we compare the average classification time, per scene, when using a simple multi-grid representation vs. F-H segmentation. In both datasets, using the F-H representation is at least 3x 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