3-D Scene Analysis via Sequenced Predictions over Points and Regions

Xuehan Xiong Daniel Munoz J. Andrew Bagnell Martial Hebert
The Robotics Institute
Carnegie Mellon University
{xxiong, dmunoz, dbagnell, hebert}@ri.cmu.edu

Abstract— **We address the problem of understanding scenes from 3-D laser scans via per-point assignment of semantic labels. In order to mitigate the difficulties of using a graphical model for modeling the contextual relationships among the 3-D points, we instead propose a multi-stage inference procedure to capture these relationships. More specifically, we train this procedure to use point cloud statistics and learn relational information (*e.g.*, tree-trunks are below vegetation) over fine (point-wise) and coarse (region-wise) scales. We evaluate our approach on three different datasets, that were obtained from different sensors, and demonstrate improved performance.**

I. Introduction

An important perception task for automated scene interpretation from a 3-D point cloud is to uniquely label each point in the scene with the semantic label of the object the point lies on which (*e.g.*, road, building, tree-trunk). Inferring the labels based only on local features is very difficult for a variety of reasons. For example, the viewpoint from which objects are perceived can widely vary, the sensor irregularly samples points from objects, and there is often local ambiguity in appearance. To address this problem, most approaches attempt to model the relationships among elements in the scene typically through a graphical model. The use of such graphical models has been widely used for both 2-D images [9], [12] and 3-D point clouds [1], [19]. Typically in these structured models, a node in the graph is a random variable representing a 3-D point's label and edges are formed to model context. In order to effectively model context, many interactions need to be considered, which result in a densely linked (loopy) graph/random field. In general, exact inference over such a random field is intractable and only approximate methods can be used. This complicates learning such models as exact inference is required and the use of approximate inference results in learned solutions that are arbitrarily bad [11]. Therefore, to mitigate this problem, in addition to restrictive interactions (*e.g.*, the prevalent Pott's model) needed for many approximate inference techniques, we propose an alternative technique for 3-D point cloud labeling that does not explicitly model a joint probability distribution.

Instead of performing inference as a generic procedure (*e.g.*, loopy belief propagation), we instead directly design and train an inference procedure via a sequence of predictions from simple machine learning modules that are structured to net the desired labeling. As detailed in Section IV-A, each module makes a *soft* prediction of what labels are present either at a point-wise scale or over a region of 3-D points; by *soft*, we mean that the classifier module outputs a probability distribution instead of a single label. The output probabilities of spatially neighboring areas are then fed as inputs into another module in order to propagate contextual information. Our contribution is how to structure these modules and use their outputs for effective 3-D scene analysis. We demonstrate the effectiveness of our approach on three different 3-D point cloud datasets obtained from three different laser scanning systems with multiple semantic labels, as illustrated in Fig. 1. The `VMR-Oakland-v2` dataset was collected from a ground vehicle scanning in profile; the `GML-PCV` dataset was collected from an airborne laser; the `RSE-RSS` dataset was collected from a ground vehicle scanning with a 360° field of view.

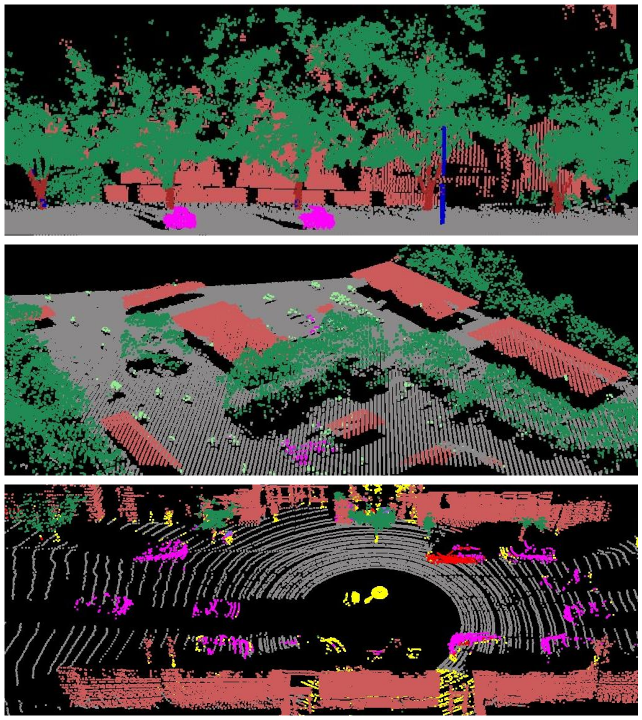
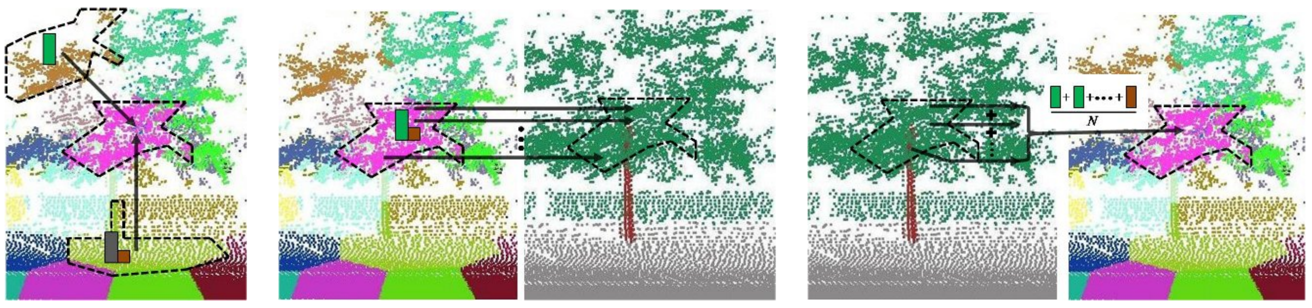


Fig. 1. Our point cloud labelings on three different datasets. Top: `VMR-Oakland-v2`. Middle: `GML-PCV`. Bottom: `RSE-RSS`. Each color indicates a different class: grey = ground, light-red = building, brown = tree-trunk, dark-green = vegetation, pink = vehicle, dark-blue = pole, light-blue = wire, light-green = shrub, dark-red = fence, yellow = background. Note: each dataset contains different subsets of these classes.



(a) (b) (c)

Fig. 2. Overview of various steps in the overall procedure. (a) shows the segmented points in the second level (colored by segment). The pink segment receives contextual information from its neighbors. (b) Contextual information is sent from the region (top level) to all points (bottom level) that constitute it (the points are colored by their class for visualization purposes). (c) Contextual information is averaged from the points and sent to the region level.

II. Related Work

3-D scene analysis has become an important problem in all types of environments, including street-level [3], [8], [7], indoor [21], [24], and aerial [16]. Patterson *et al.* approached this problem by memory-based learning [20]. During training, point and object features are computed and stored in databases A and B. During inference, points are classified as positive or negative samples through nearest neighbors in database A. Then, they connect the neighboring positive sample points to form segments. The database B is queried and returns a positive or a negative detection based on the thresholded distance between the stored models and each extracted segment. In addition to inference time as a function of the database size, such an approach is also sensitive to the detected regions of interest and selected threshold value. Lai and Fox [13] also investigate a similar data-driven technique using sampled CAD models downloaded from the internet; we compare our approach with this in Section VI-C. In contrast with the above exemplar based approaches, our algorithm does not require ground removal or such pre-filtering of the point cloud and uses contextual information. Other models that follow a similar statistical learning approach often use graphical models [1], [19], [22]. We compare with two different types of graphical models in our analysis (Sec. VI) to illustrate our approach's effectiveness.

III. Overview

The main idea of our approach is that we directly train the steps of an inference procedure to ultimately obtain a semantic per 3-D point labeling. Our approach builds off the work of Munoz *et al.* [18] for 2-D scene analysis. The following describes their procedure at a high-level, which will become more clear when we detail our procedure for the 3-D case in the following paragraphs. In [18], the image is analyzed by examining multiple regions from coarse to fine via a hierarchical segmentation of the image. Starting at the top level in the hierarchy, their algorithm trains a first classifier to predict the label distributions present in the level's regions. Next, a second classifier is again trained over the level's regions but now also uses the first classifier's predictions from neighboring regions to encode context, *i.e.*, the second classifier is sensitive to the context within the level. The outputs from the second classifier are then passed to the child level which are used in conjunction with the child regions' image features to train another classifier for the finer regions. This refining process is repeated until the bottom level is reached.

Figure 2 illustrates the analogous steps of [18] in the 3-D domain. We construct a two-level hierarchy[1] where the bottom level consists of points and the top level is a segmentation which may contain regions that consist of mixed labels. The regions in the hierarchy do not change during the procedure. In practice, we use k-means++ [2] clustering over (x, y, z), where the k is defined to be 1% of the number of points, to produce segments that (roughly) cover the non-ground objects. Without the ambiguities introduced from a 3-D to 2-D projection in images, we observe that this simple clustering produces more stable segmentations than is typically observed with sophisticated image segmentations. Furthermore, our 3-D features are less sensitive to the segmentation quality than is needed to model shape in images.

In Fig. 2(a) we illustrate the use of contextual information for regions in the top-level. At this stage, each region in the level is already associated with a predicted label distribution from the previous classifier. We update the *intra-level context* of each region with features that encode the neighboring regions' predictions, as illustrated with the pink region using its neighbors. In Section IV-B we investigate how to effectively encode these 3-D relationships. Another classifier is then trained across all regions in this level, where each region's features consist of point cloud descriptors and the intra-level context information. The main idea is that the contextual information should refine the prediction. For example, the partially predicted tree-trunk below the pink region should help decipher that it is mostly vegetation; in Section IV-B we demonstrate that our approach learns this naturally occurring phenomenon.

In Fig. 2(b), the newly predicted label distribution is then passed to the points that constituted the region. These predictions are used as one source of *inter-level context* for each point. Now, a classifier is trained over the points that uses this inter-level context in addition to the point cloud

[1]We observed marginal difference in performance beyond this two-level hierarchy but note that our description naturally extends to the deeper case.

descriptors. As similarly done in (a), we also train another classifier at the point-level that uses neighboring points' predictions. In addition to parsing down the hierarchy, we can similarly proceed up the hierarchy. In Fig. 2(c), the predicted label distributions of the points belonging to region are averaged and sent to it. These averaged distributions are the other inter-level context of a region. The number of times we go up/down the hierarchy and the number of rounds we perform intra-level context updates in a level are determined by a validation set.

IV. Training the Parsing Procedure

For the remainder of the paper, we will interchangeably call "region" both a group of 3-D points in the top level or a single 3-D point in the bottom level. The next subsections describe the individual components of the learning procedure. We first explain our choice of the base classifier used to classify regions (Sec. IV-A). We then describe how we use neighboring predictions to encode context. Using these two modules, we describe how we use intra-level context through a sequence of classifiers (Sec. IV-C) and how they are tied together up and down the hierarchy (Sec. IV-D).

A. Base Classifier

We define C_i to be the random variable representing the class of a region, $x_i \in \mathcal{R}^d$ to be its features (note that this will include both point cloud descriptors and contextual information), and y_i to be its ground truth distribution of K labels, *i.e.*, y_i is a vector of length K that contains values in $[0, 1]$ and sums to 1. We use a simple K-class logistic regression (LogR) model Q_w, parameterized by $w \in \mathcal{R}^{Kd}$ to predict the label distributions per region:

$$Q_w(C_i = k|x_i) = \frac{\exp(w_k^T x_i)}{\sum_{a=1}^{K} \exp(w_a^T x_i)}. \quad (1)$$

In order to handle regions that may contain more than one label, we train the model by minimizing the cross entropy of y_i and the model Q_w; this reduces to a weighted max-likelihood estimation problem:

$$\arg\max_{w} \sum_i \sum_k y_{i,k} \log Q_w(C_i = k|x_i) - \lambda ||w||^2, \quad (2)$$

where $y_{i,k} = y_i[k]$ and $\lambda > 0$ regularizes the model. This problem is concave and we optimize it with stochastic gradient ascent.

Given a training set of samples $X = \{x_i\}$ and respective distributions $Y = \{y_i\}$, we denote $w = \mathcal{T}(X, Y)$ as solving the maximization problem in Eq. 2. Given a set of samples X and parameters w, we denote $\widehat{Y} = \mathcal{P}(X, w)$ as predicting the label distributions (Eq. 1) for each sample.

B. Contextual Features

Given that each region in a level is associated with a distribution of labels, we wish to model the neighboring spatial context that surrounds each region. To do this, we compute contextual features that encode these spatial distribution of labels. These context features can then be appended to a region's feature vector x_i and used in the classifier. In our approach, we model what is above, below, and adjacent to each region, as illustrated in Fig. 3. That is, we construct a neighborhood sphere centered at a region's centroid and then equally divide this sphere into three slices along the z-axis (vertical direction). In the bottom and top levels we use large radii of 4 m and 12 m, respectively. Large radii are needed to cover the large spatial extent from an object's centroid to its neighboring regions' points. Within each slice we average the label distributions of the contained points to obtain a feature vector of length K, per slice. In order to model the spatial configuration of the neighboring points, (*e.g.*, neighboring points on poles should lie directly above/below) within each slice we also average of the angles formed between the positive z-axis and the vector from the region's centroid O to each neighboring point N_i, resulting in a value in $[0, \pi]$.

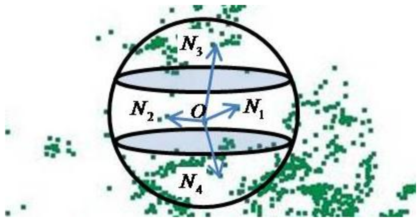


Fig. 3. Illustration of intra-level contextual features.

Given a set of features X and label distributions Y, we denote the process of computing and appending/updating the $3(K + 1)$ contextual features to each x_i as $X' = \phi(X, Y)$.

C. Multi-Round Stacking (MRS)

By sequentially training a series of classifiers, we can ideally learn how to fix the mistake from the previous one. In addition, we can use these previous predictions as contextual cues. That is, given a labeled training set X, Y, we first train a classifier $w^1 = \mathcal{T}(X, Y)$ over the entire training set. Using w^1, we can classify X to generate predictions $\widehat{Y} = \mathcal{P}(X, w^1)$ from which to derive contextual cues, as described above, and then train a new classifier w^2. Specifically, we use $\widehat{Y}$ to create a new feature set with contextual cues $X^1 = \phi(X, \widehat{Y})$ and train a new classifier $w^2 = \mathcal{T}(X^1, Y)$. The process can be repeated for multiple rounds until no improvement is observed. However, note that if we were to use w^1 to classify our training data, the resulting predictions would be more optimistically correct than it would be on the unseen test data. Therefore, training another classifier on the outputs is prone to overfitting. To mitigate this issue, we use the technique of *stacking* [23] as successfully used in [5], [18]. The following details how we use stacking in the context of our application; the procedure is similar to cross validation.

Instead of training a single classifier over the entire training set, we generate multiple temporary classifiers that are solely used at training time. The purpose is to generate predictions over the examples that were not used to train a classifier. We equally partition the training set into 5 disjoint subsets $X = \{X_i\}_{i=1}^5$ and $Y = \{Y_i\}_{i=1}^5$. For each

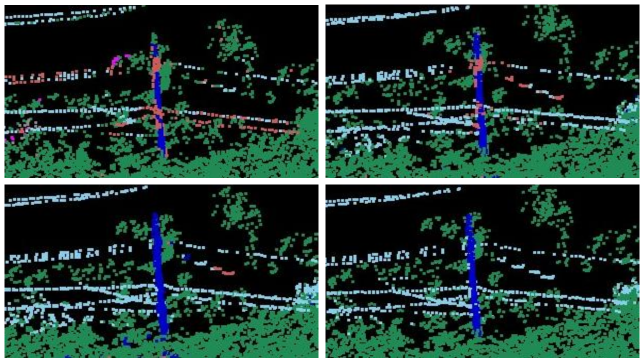


Fig. 4. Classification evolution via multiple rounds of stacking. Top-left: initial classification, Top-right: 1 round Bottom-left: 2 rounds, Bottom-right: 4 rounds.

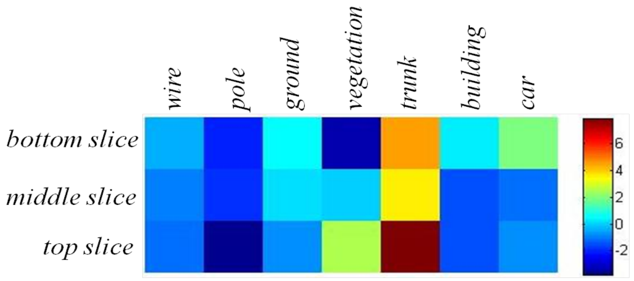


Fig. 5. Learned *tree-trunk* weights for its contextual features.

subset, we train a temporary classifier on the other subsets $\gamma = \mathcal{T}(X - X_i, Y - Y_i)$ and then use γ only on the samples in X_i to generate predictions $\widehat{Y}_i = \mathcal{P}(X_i, \gamma)$. Note that this procedure is only done at training time to generate the predictions $\widehat{Y}$ and each temporary classifier γ is discarded afterwards. The intuition behind this technique is that the context derived from $\widehat{Y}$ is as "noisy" during training as it will be during testing.

In [18], only one iteration of stacking is used per level; however, we found it beneficial to do multiple rounds of stacking (MRS). Figure 4 illustrates the effectiveness of performing multiple rounds of stacking on real data; the scene is a closeup of a utility pole near vegetation. Initially, the classification is poor as many of the *wires/power-lines* are mistaken as *building*, but the contextual cues help the subsequent rounds. At each round, we update each region's context features from the previous round, instead of always concatenating. Regions that do not fit in the context are iteratively corrected.

It is useful to verify that we learn meaningful context. Therefore we examined our learned parameters w to observe the types of relationships we learned. In Fig. 5, we plot the portion of $w_{tree-trunk}$ that corresponds to the contextual features. Here we observe that the algorithm correctly learned that a *tree trunk* region is likely to have vegetations above but not below and to have *car* and *ground* in the bottom slice but not in the top slice.

D. Stacked 3-D Parsing

In this section we tie all the previous components together, resulting in a 3-D parsing algorithm that uses three sources of contextual information: points, regions, and spatial context through stacking. We refer to this entire algorithm as Stacked 3-D Parsing (S3DP).

Given a labeled point cloud, we construct the two-level hierarchy, extract point cloud feature descriptors, and create ground truth label distributions for both the bottom (X_b, Y_b) and top (X_t, Y_t) levels. We then determine in which order the hierarchy should be traversed. The number of traversals and rounds of stacking that we perform is found through validation. In practice, we often obtain good performance with starting at the bottom, going up, and then back down.

The following describes how we parse **up** the hierarchy:

1) Apply N rounds of MRS on a dataset (X_b, Y_b). MRS returns a sequence of $N+1$ classifiers $\mathbf{f_{(b)}} = \{w^n_{(b)}\}^{N+1}_{n=1}$ and the hold-out predictions $\widehat{Y}_b$ from the last round of stacking.
2) Extend each **region** feature vector $x_i \in X_t$ with **the average of its children's** probability distributions in $\widehat{Y}_b$.
3) Apply N rounds of MRS on a dataset (X_t, Y_t), which returns $\mathbf{f_{(t)}} = \{w^n_{(t)}\}^{N+1}_{n=1}$.
4) Save $\mathbf{f}_{(t)}$ and $\mathbf{f}_{(b)}$ for inference.

The following describes how we parse **down** the hierarchy:

1) Apply N rounds of MRS on a dataset (X_t, Y_t). MRS returns a sequence of $N+1$ classifiers $\mathbf{f_{(t)}} = \{w^n_{(t)}\}^{N+1}_{n=1}$ and the hold-out predictions $\widehat{Y}_t$ from the last round of stacking.
2) Extend each **point** feature vector $x_i \in X_b$ with **its parent** probability distributions in $\widehat{Y}_t$.
3) Apply N rounds of MRS on a dataset (X_b, Y_b), which returns $\mathbf{f_{(b)}} = \{w^n_{(b)}\}^{N+1}_{n=1}$.
4) Save $\mathbf{f}_{(t)}$ and $\mathbf{f}_{(b)}$ for inference.

Inference proceeds in the same sequence, where we use the classifiers in $\mathbf{f}$ to make predictions in the same order in which we trained the procedure.

V. Experimental Setup

A. Features

At the bottom (point) level of the hierarchy we compute three geometric features [17], [15]. These values measure scatter, linearity and planarity of the local neighborhood of points within a specified radius. In our experiments, the neighborhood is defined within a 0.8 m radius for the `VMR-Oakland-v2` and `RSE-RSS` datasets and 2 m for the `GML-PCV` dataset. We also use two directional features to capture the local orientation. These are the scalar projection of the locally estimated tangent and normal directions onto the z-axis.

The following features are used both in the bottom and top levels: the dimensions of the bounding box that encloses the points in the three principal component space, spin images [10] around z-axis, and relative elevations. The bounding box is computed over the local neighborhood in the bottom level and over the regions themselves in the top level. Spin images are 5×5 in the bottom level and 11×11 in the top level, with each cell being 0.2 m $\times$ 0.2 m. To estimate relative elevations, we first compute a 2.5-D elevation map with 10 m $\times$ 10 m cells that contain the min and max z-coordinates (elevation). We then compute the two differences in elevation between the region's centroid elevation and its cell's two extrema.

In the top level we compute two additional types of features. The first computes the differences between each region's centroid and its min and max elevations. The second evenly breaks the region into two components along the z-axis and then computes a 10 bin histogram over the bottom-level features from the points in each component.

B. Evaluation Metric

To compare the results from different algorithms we consider the average of the per-class F_1 scores. The F_1 score of a class k is the harmonic means of its precision p_k and recall r_k and is defined as $2p_kr_k/(p_k + r_k)$. We use F_1 instead of overall accuracy as the latter can hide poor performance of classes with few samples, and we observe a class imbalance in all the datasets we analyze. In `VMR-Oakland-v2` dataset, we generate 5 different training and testing splits and average the F_1 scores across all splits. For the `GML-PCV` and `RSE-RSS` datasets, we follow the same respective evaluation methods used in [22] and [13]. In addition, for each dataset we also show the results obtained by our base classifier LogR trained using only local point features.

VI. Experimental Results

A. `VMR-Oakland-v2` Dataset

This dataset[2] of the area surrounding the Carnegie Mellon University campus contains 3.1 M points that were scanned in profile from a laser at ground level. We divide the data into 36 sets, each of which is contains about 85,000 points, to facilitate training (6 sets), validation (6 sets) and testing (remaining sets). The objects we are interested in are *wire*, *pole*, *ground*, *vegetation*, *tree-trunk*, *building*, and *car*. We compare with the linear, associative Max-Margin Markov Network (M3N) approach[3] of Munoz *et al.* [19]. Both algorithms use the same training, validation, and testing sets as well as the same features described above (except the M3N cannot use the contextual features). In the M3N, the nodes are defined over the points, edges are constructed using 5-NN, and higher order cliques are our same top level regions. The edge features are the inverse differences of the neighboring nodes' features.

Our method achieved an average F_1 score of 0.76 while M3N and LogR average 0.71 and 0.61, respectively. Qualitative results of M3N and S3DP are shown in Fig. 6. We can see a significant improvement on the points that are at the boundary of object regions using S3DP. This can be explained by the way M3N represents interactions between points. The higher order potentials over regions prefer for the region to have a homogeneous label. In contrast, our approach supports heterogeneous label distributions and it can preserve the correct labeling at the boundary of regions. For example, in (f) the top border of a building is misclassified as *vegetation* using M3N model. This is because, the points possess similar local features (scatter and high elevation) that resemble *vegetation*. Furthermore region-wise features do not disambiguate the problem and the high-order clique will prefer to group these points as the same incorrect label. S3DP solved this problem by learning the context of the regions in the top level. Such context becomes essential when local features and point context both failed to capture the correct label. The shrubs in the bottom-left corner of the scene illustrate a situation when M3N succeeds (e) while S3DP fails (f). S3DP confuses shrubbery *vegetation* with *building* because the algorithm learned to "correct" *vegetation* into *building* since it fits into the context more properly (*building* regions are more likely to be directly above another *building* region than a *vegetation* region). M3N ignores the relational context above and correctly classifies this region based on its local features.

Table I shows individual class performance of the three algorithms. As we expected, LogR performs the worst of the three since it does not include any contextual information. From its poor performance on *tree-trunk, car*, and *wire* we can see that this is a difficult dataset. S3DP is superior in classifying car and tree trunk. Illustrated in Fig. 5, our algorithm learned that a narrow linear region with vegetation above is more likely belong to *tree-trunk* than *pole*. In 3-D data without the color and textural information, this learned relational feature has become the only difference between narrow tree trunks and poles. M3N fails to capture such spatial layout of different objects (Fig. 6-d).

		wire	pole	ground	veg	trunk	bldg.	vhc.
P	S3DP	**0.73**	0.51	**0.99**	**0.96**	**0.65**	**0.83**	**0.79**
	M3N	0.66	**0.55**	**0.99**	0.94	0.55	0.80	0.70
	LogR	0.49	0.42	**0.99**	0.90	0.46	0.74	0.63
R	S3DP	**0.75**	**0.67**	0.98	0.93	**0.41**	**0.93**	**0.74**
	M3N	0.72	0.63	**0.99**	**0.94**	0.30	0.92	0.43
	LogR	0.52	0.53	**0.99**	0.90	0.13	0.87	0.38

TABLE I
`VMR-Oakland-v2` precisions (P) and recalls (R), averaged over five splits.

The following demonstrates that contextual information from three directions all contribute in the classification task. In an inference procedure, let (N, S) indicate the number of intra-level stacking rounds N and $S \in \{\uparrow, \downarrow\}^h$ be a parsing sequence of h traversals, where $\uparrow$ and $\downarrow$ indicates traversing up and down the hierarchy, respectively. For one split on validation data we scored 0.72 for $(N, S) = (0, \downarrow)$, 0.79 for $(1, \downarrow)$, 0.81 for $(2, \downarrow)$, and 0.82 for $(1, \uparrow\downarrow)$. Such configurations are performed for all datasets and the best performing configuration on validation data is selected.

[2] http://www.cs.cmu.edu/~vmr/datasets/
[3] http://www.cs.cmu.edu/~vmr/software/software.html

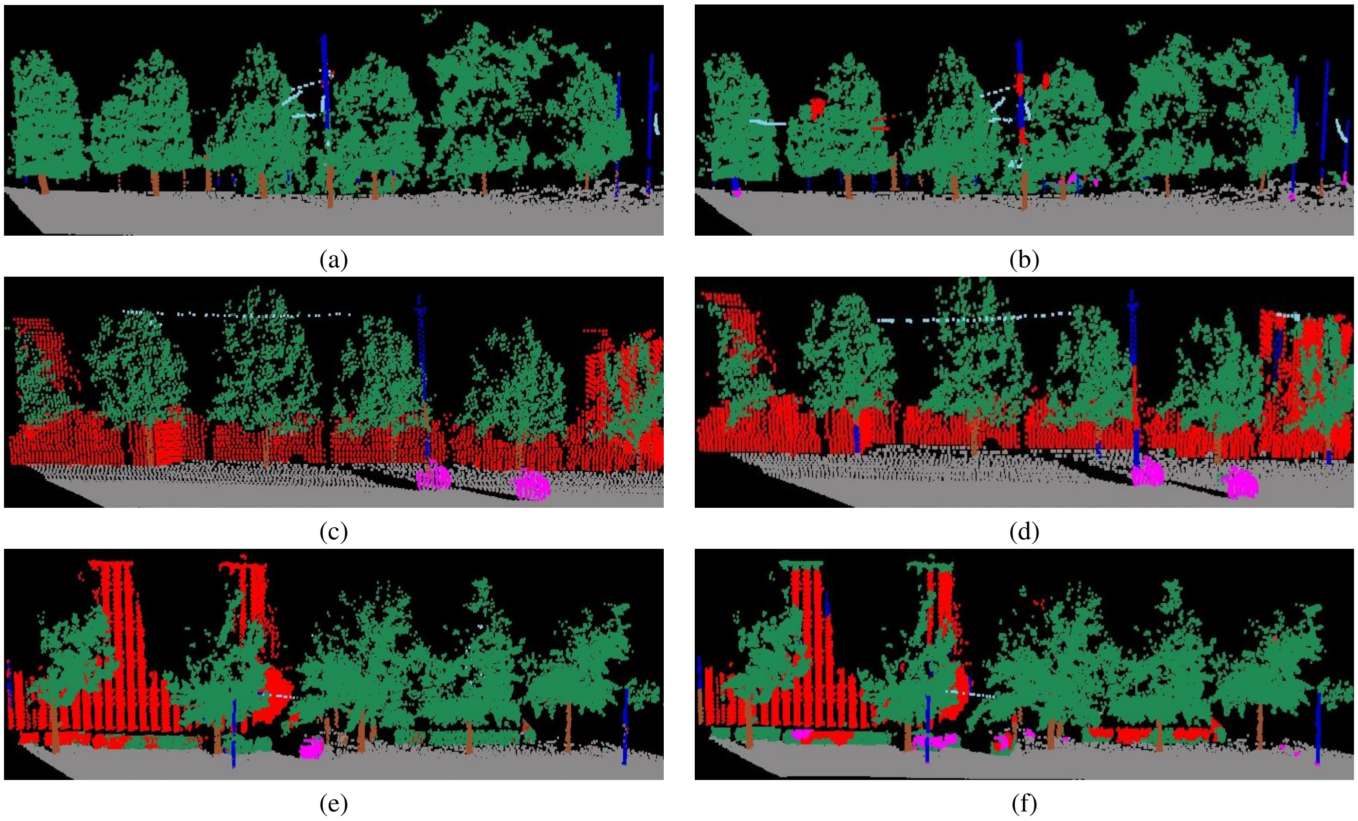


Fig. 6. Example classifications from `VMR-Oakland-v2` dataset using S3DP and M3N are shown in the left column and right column, respectively. Buildings are colored by dark red to increase the contrast between building and tree trunks.

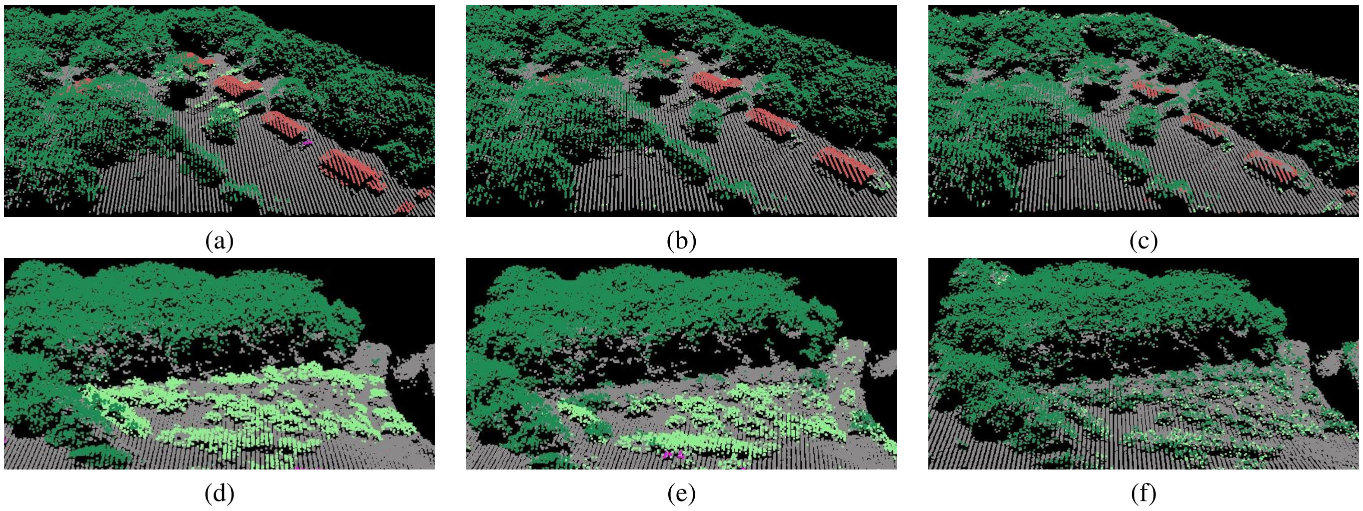


Fig. 7. Example classifications from `GML-PCV`. Left column: ground truth labeling. Middle column: S3DP classification. Right column: LogR classification.

B. `GML-PCV` Dataset

We evaluated our algorithm on an aerial dataset[4] from Shapovalov *et al.* [22]. The evaluation is performed on two different datasets A and B. Each dataset contains a training and testing split of about 1 M points each. We divide each training subset into two for parameter learning and validation. Set B contains four classes (*ground*, *roof/building*, *tree*, *low vegetation/shrub*) and set A contains the same four and *cars*. The two sets are evaluated separately, as done in [22]. Shapovalov *et al.* [22] use a pairwise Markov network that is constructed over segments instead of points. However, instead of associative/Pott's potentials, they model the co-occurrence relationship between classes. They refer to this model as a Non-Associative Markov Network (NAMN).

Table II shows the per-class performance of the three algorithms. Our algorithm outscored NAMN 0.66 to 0.59 in dataset A and 0.85 to 0.77 in dataset B. LogR averages

[4]http://graphics.cs.msu.ru/en/science/research/3dpoint/classification

	Dataset A	ground	bldg	tree	low veg	car
P	S3DP	**0.95**	**0.91**	**0.99**	0.31	**0.54**
	NAMN	0.90	0.87	0.92	**0.72**	0.37
	LogR	0.92	0.74	0.96	0.06	0.03
R	S3DP	**0.98**	**0.77**	0.98	**0.36**	0.10
	NAMN	0.96	0.58	**0.99**	0.09	**0.16**
	LogR	0.96	0.37	0.93	0.13	0.01
	Dataset B					
P	S3DP	**0.99**	0.83	**0.97**	**0.53**	
	NAMN	**0.99**	**0.88**	0.95	0.25	
	LogR	0.98	0.79	0.88	0.38	
R	S3DP	**0.99**	**0.92**	**0.97**	0.52	
	NAMN	0.98	0.81	0.89	**0.57**	
	LogR	**0.99**	0.63	0.96	0.10	

TABLE II

GML-PCV DATASETS A AND B PRECISIONS (P) AND RECALLS (R).

0.49 in A and 0.69 in B. In A all three algorithms perform poorly on *cars* and *low vegetation*. In contrast with VMR-Oakland-v2, *car* objects include much fewer points. The shape information of small objects is almost lost in these long range aerial scans. Because of this, *car* and *low vegetation* share similar local features and no contextual information can be used to differentiate between the two. Without the confusion between *cars* and *low vegetation* we can see an improvement on *low vegetation* in dataset B.

One key characteristic of this dataset is that the *ground* points are not on the same elevation, so the elevation of a point provides little information about its class. Due to such difficulty, LogR has extremely poor performance on *low vegetation* and *car*. For example, LogR cannot distinguish *low vegetation* from *high vegetation* because of their similar local features (Fig. 7-f). Our algorithm used stacking to learn that *low vegetation* has a high distribution of *ground* in its neighborhood while *high vegetation* does not and corrects this mistake (Fig. 7-e).

Another example is shown in Fig. 7-c. Using only local features, points on the *roofs/building* are confused with *ground* because when there is a large area of *building*, the relative 2.5-D elevation feature can no longer discriminate between *ground* and *buildings*. Our algorithm learns that *building* regions are above the neighboring *ground* and propagates this information. It corrects mislabeled *building* points using this contextual information (Fig. 7-b).

C. RSE-RSS Dataset

This dataset[5] from Lai and Fox [13] contains 10 scans of approximately 65,000 points each. It was collected by a Velodyne laser scanner on the ground. We consider it as the most difficult dataset of the three due to the noisy and sparse point measurements as well as noisy ground truth labelings. Our task is to classify each point into the following 8 classes: *ground*, *street sign*, *tree*, *building/house*, *fence*, *person*, *car*, and *background* (which is everything else). We follow the evaluation method described in [13] where we randomly generate 10 training and testing splits and average the results from them. In each training split, we use 5 files for training and use leave-one-out cross-validation to select parameters.

[5]http://www.cs.washington.edu/homes/kevinlai/datasets.html

In [13], the authors perform segmentation over the point cloud and then classify each resulting segment using a non-parametric method. In order to obtain better segmentation, they apply ground removal to the scene beforehand. A distance function is associated with every segment and it is learned by maximizing the margin between the segments of the same class and the segments from other classes. They also introduced a *Domain Adaptation* (DA) method to learn with CAD model data in conjunction with the real point cloud data. They demonstrated increased performance compared with only training on real scans.

The *background* class could contain various objects such as stairs, fire hydrants, and traffic lights. Treating them as a single, broad class eliminates learning context among different classes that could be helpful in classification. For example, stairs often appear with buildings but by combining stairs, fire hydrants and traffic lights into a single background class makes learning such relations prohibited. Furthermore, modeling such a broad class that mixes objects of drastically different appearance is hard with our linear classifier. In addition, Fig. 8-c shows *trees*, *ground* and *houses* that are mislabeled as *background*. As a result, our algorithm learns that *background* can appear in the neighborhood of any other class. The above issues explain why stacking does not fix the mislabeling between background and other classes (Fig. 8-a).

In Table III, our algorithm demonstrates comparable results with using only real scan data. Our F_1 scores are from the confusion matrix over all 8 classes while the DA result from [13] is over 7 classes without ground; performance on background is not reported in their paper. Our algorithm averages 0.46 over the remaining 6 classes while DA scores 0.47. Our algorithm shows its advantage on the classes that can provide enough training samples. The better performance DA on classes with fewer examples, such as *person* and *car*, may be due to the use of additional training data from the CAD models. A natural extension in future work would be to add domain adaptation to our algorithm, such as using the simple technique from Daume III [6].

bkgd	st sign	ground	tree	house	fence	person	car
0.79	0.28	**0.94**	**0.66**	**0.83**	0.31	0.20	0.49
N/A	**0.35**	N/A	0.55	0.73	**0.32**	**0.31**	**0.57**
0.78	0.21	**0.94**	0.51	0.74	0.17	0.13	0.29

TABLE III

RSE-RSS PER-CLASS F_1 SCORES. TOP ROW: S3DP. MIDDLE ROW: DOMAIN ADAPTATION [13]. BOTTOM ROW: LOGR.

D. Timings

We analyze the run-time behaviors of S3DP and M3N on the VMR-Oakland-v2 dataset using an Intel Xeon X5670 2.93 Ghz processor. Both S3DP and M3N use the same configuration for constructing the regions and feature computations. On average over a test point cloud, k-means and feature computation take 5.4 s and 11.2 s, respectively.

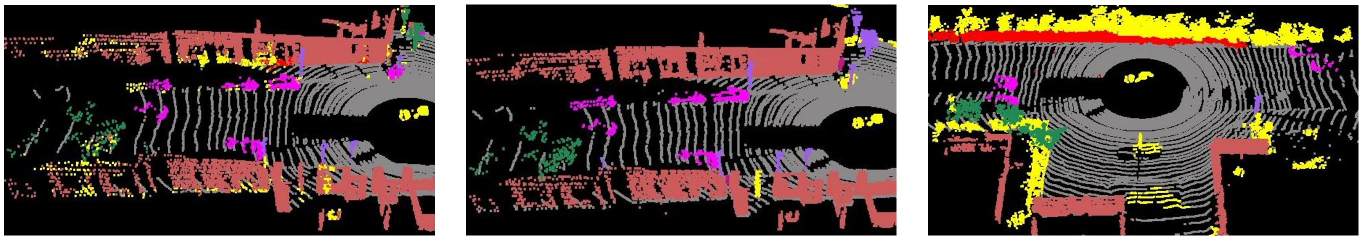

(a) (b) (c)

Fig. 8. Example classifications from RSE-RSS. (a) S3DP classification. (b) Ground truth labeling. (c) Example of noisy ground truth mislabeling.

With S3DP, a preprocessing step is done to compute the neighbors from which the contextual features (Fig. 3) are computed, and then inference is performed with a sequence of predictions. With M3N, the high-order random field structure uses the same regions as in S3DP, but additionally constructs edge potentials in the random field and then solves multiple min-cut/max-flow problems for inference.

With S3DP, performing one traversal sequence takes linear time in the number of regions; as mentioned before, we found that using two traversals often works well. On average, performing 5 sequential predictions that traverses up and down the hierarchy totals 2.4 s. With our random field structure that locally links neighbors, performing a single min-cut operation takes time roughly quadratic in the number of regions. On average, performing graphcut inference totals 10.4 s, using the optimized implementation from [4].

The time complexity for the feature computation and the preprocessing steps depend on the neighborhood search algorithm. In our implementation, we used a simple k-d tree to perform spatial range searches in the point cloud and parallelized these lookups over 8 cores when possible. Computing the neighborhoods for the contextual features totals 19.3 s, and linking 5-NN in the random field totals 2.9 s. In this work we analyzed point clouds assuming an unorganized structure and thus the majority of our computation is due to performing range searches and is not due to expensive computations inherently needed by the inference algorithm. Future work will aim to decrease computation time inherent to 3-D processing by taking advantage of structured readings (*e.g.*, ordered scan lines), faster data structures (*e.g.*, [14]), and focused processing over smaller areas of interest.

VII. Conclusion

In this work we propose a sequential parsing procedure for scene analysis in 3-D point clouds. This procedure performs a series of very simple predictions and can effectively encode neighboring context. In our analysis, we demonstrate that it can learn meaningful spatial layout of objects such as tree-trunks are below vegetation. Over an extensive set of experiments, we demonstrate that this conceptually simple approach is favorable in many environments that are scanned with different sensors.

Acknowledgements

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. D. Munoz was supported by a QinetiQ North America Robotics Fellowship.

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 3-d scan data. In *CVPR*, 2005.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, 2007.

[3] J. Behley, K. Kersting, D. Schulz, V. Steinhage, and A. B. Cremers. Learning to hash logistic regression for fast 3d scan point classification. In *IROS*, 2010.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *T-PAMI*, 26(9), 2004.

[5] W. W. Cohen and V. R. Carvalho. Stacked sequential learning. In *IJCAI*, 2005.

[6] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, 2007.

[7] B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, and S. Singh. A pipeline for the segmentation and classification of 3D point clouds. In *ISER*, 2010.

[8] O. Hadjiliadis and I. Stamos. Sequential classification in point clouds of urban scenes. In *3DPVT*, 2010.

[9] X. He, R. S. Zemel, and M. A. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[10] A. E. Johnson and M. Hebert. Using spin-images for efficient multiple model recognition in cluttered 3-D scenes. *T-PAMI*, 21(5), 1999.

[11] A. Kulesza and F. Pereira. Structured learning with approximate inference. In *NIPS*, 2007.

[12] S. Kumar and M. Hebert. Discriminative random fields. *IJCV*, 68(2), 2006.

[13] K. Lai and D. Fox. 3D laser scan classification using web data and domain adaptation. In *RSS*, 2009.

[14] J.-F. Lalonde, N. Vandapel, and M. Hebert. Data structure for efficient dynamic processing in 3-d. *IJRR*, 26(8), 2007.

[15] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10), 2006.

[16] W. L. Lu, K. Okuma, and J. J. Little. A hybrid conditional random field for estimating the underlying ground surface from airborne lidar data. *IEEE T-GRS*, 47(8), 2009.

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

[18] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *ECCV*, 2010.

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

[20] A. Patterson, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3-D datasets using bottom-up and top-down descriptors. In *ECCV*, 2008.

[21] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11), 2008.

[22] R. Shapovalov, A. Velizhev, and O. Barinova. Non-associative markov networks for 3D point cloud classification. In *Photogrammetric Computer Vision and Image Analysis*, 2010.

[23] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2), 1992.

[24] X. Xiong and D. Huber. Using context to create semantic 3D models of indoor environments. In *Proc. BMVC*, 2010.