2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Simplified Markov Random Fields for Efficient Semantic Labeling of 3D Point Clouds

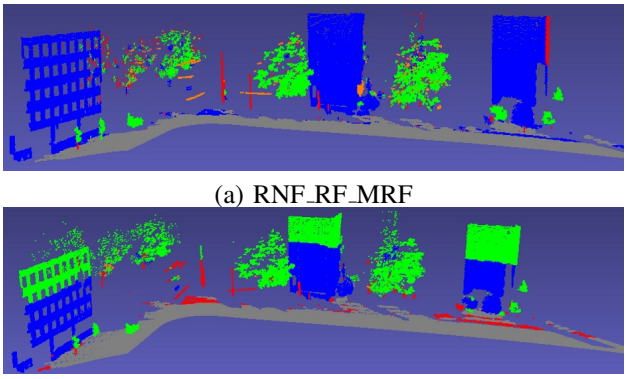
Yan Lu and Christopher Rasmussen

Abstract— **In this paper, we focus on 3D point cloud classification by assigning semantic labels to each point in the scene. We propose to use simplified Markov networks to model the contextual relations between points, where the node potentials are calculated from point-wise classification results using off-the-shelf classifiers, such as Random Forest and Support Vector Machines, and the edge potentials are set by physical distance between points. Our experimental results show that this approach yields comparable if not better results with improved speed compared with state-of-the-art methods. We also propose a novel robust neighborhood filtering method to exclude outliers in the neighborhood of points, in order to reduce noise in local geometric statistics when extracting features and also to reduce number of false edges when constructing Markov networks. We show that applying robust neighborhood filtering improves the results when classifying point clouds with more object categories.**

I. INTRODUCTION

As robot navigation tasks become more complicated and challenging, range finding devices that can capture 3D point clouds of the environment start to be standard equipments for mobile robot systems. Point clouds contain valuable geometric information of scenes which is ambiguous and unreliable if otherwise recovered from single image or stereo images. Therefore, automated point cloud interpretation is important for autonomous robot navigation tasks such as obstacle avoidance, object detection, and environment modeling. Processing 3D point clouds is by no means easier than processing 2D images, even though the former circumvents the ambiguity induced by 3D to 2D projection. Unlike images that can be naturally represented as intensity or color functions explicitly defined over spatial locations, geometric information contained in 3D point clouds is implicit and purely represented by the spatial arrangement of the observed points. Additionally, laser sensors irregularly sample points from objects and the viewpoint from which objects are perceived can largely vary, and thus, 3D point clouds are usually sparse and locally ambiguous in geometric appearance. In order to solve these problems, utilizing contextual information and modeling contextual relations is always essential for point cloud processing.

In this paper, we focus on 3D point cloud classification by assigning semantic labels to points in the scene. We first extract features that incorporate both local geometric statistics and global contextual information of the point cloud. We then use simplified Markov networks to model the contextual relations between points, where the node potentials are calculated from point-wise classification results using standard classifiers, such as Random Forest (RF) and Support Vector Machines (SVM), and the edge potentials are set by physical distance between points. In addition, we apply robust neighborhood filtering for excluding outliers in the neighborhood of points, in order to reduce noise in local geometric statistics when extracting features and also to reduce the number of false edges when constructing Markov networks. Fig. 1 shows snapshots of labeling results[1] of the CMU Oakland 3D point cloud data set obtained from our method RNF_RF_MRF and M^3N (Max Margin Markov Random Networks) proposed in [1], respectively.



(a) RNF_RF_MRF

(b) M^3N [1]

Fig. 1. Example classification results of the CMU Oakland 3D point cloud data set. (a) Results obtained from our method RNF_RF_MRF. It performs robust neighborhood filtering (see Section IV for explanation), and uses classification results obtained from a random forest classifier for calculating node potentials of Markov networks (see Section III for details). (b) Reproduced results obtained from M^3N proposed in [1]. Color scheme: green-vegetation, orange-wire, red-pole/trunk, gray-ground, blue-facade.

In previous work [2][3][4][5], formulating point cloud classification problem within a MRF or Conditional Random Field (CRF) framework has proven to be an appropriate choice because it utilizes contextual information to produce improved classification results over locally independent classifiers. In this paper, instead of learning weight vectors for calculating node and edge potentials of Markov networks, we obtain node potentials from point-wise classification results, and obtain edge potentials directly from the physical distance between points. Classifying new data using our simplified MRF is significantly faster than methods that learn node and edge weights for constructing Markov networks, such as M^3N [1], and timing is an important factor when

The authors are with the Dept. of Computer & Information Sciences, University of Delaware, Newark, DE, USA. Their e-mail addresses are yanlu@udel.edu and cer@cis.udel.edu, respectively.

[1]This paper is best viewed in color.

considering running a method on board. Another contribution of this paper is that, in the phases of feature extraction and Markov network construction, we propose a novel robust neighborhood filtering method to exclude outliers in the neighborhood of a point. By applying such filtering, we reduce noise in local geometric statistics and eliminate false edges in Markov networks that cause over smoothed-out effect after minimizing energy for Markov networks. We show in Section V-B that robust neighborhood filtering improves final classification results.

This paper is organized as follows. We first review related work in Section II. Then, we describe in Section III the simplified MRF framework, including the way we construct Markov networks and the way we calculate node and edge potentials. In Section IV, we explain in detail our robust neighborhood filtering method for excluding outliers in the neighborhood of a point. In section V, we show, compare, and discuss the classification results obtained from our method and state-of-the-art methods. Finally in section VI, we conclude and propose the future work.

II. Related Work

One way of 3D scene interpretation is through classification, where each 3D point is assigned a unique label that can either be conceptual (scatter, linear, surface, etc.) [6] or semantic (ground, building, vegetation, etc.) [2][1][7]. We have seen point cloud classification for various types of environments, such as urban [4][1][8], naturally unstructured [6][9], indoor [5][10], and aerial [7]. Some work process directly on the original point cloud [2][11], and the others over-segment the cloud and take voxel or supervoxel as the processing element for efficiency purposes [4][12][5]. Spin images [13], spectral and directional features [12], height distribution along vertical columns, etc., have been used as geometric features.

Supervised learning methods are usually adopted for point cloud classification. Lalonde *et al.* [6] train a Gaussian mixture model using the expectation maximization algorithm, and predict new data with a Bayesian classifier. In [9], the authors presented a two-stage approach for identifying ground points from point clouds, with the first stage filtering out non-ground points based on local height and the second stage applying SVM on a set of geometric features to identify which of the remaining points belong to the ground. Lai *et al.* [8] propose a data-drive technique that leverages data sets available on the world wide web for training in order to reduce the need for manually labeled training data.

Inferring labels solely based on local features as in [6][9] is very difficult because it ignores spatial context information. To address such a limitation, most approaches attempt to model relations between points, typically through a graphical model such as CRF or MRF, and infer point labels using energy minimization algorithms such as graph cuts and belief propagation. Anguelov *et al.* [2] and Munoz *et al.* [1] apply max-margin learning to obtain node and edge weight vectors for constructing a Markov network. Lim *et al.* [4] use maximum a posteriori estimation to obtain the weights for constructing a conditional random field. In [3], the authors extend the popularly used Associative Markov Network (AMN) to learn directionality in clique potentials, and such an anisotropic model is aimed for better classification of finer objects such as poles and power lines in point clouds. Going beyond directional AMN, Shapovalov *et al.* [7] propose non-associative Markov networks for classification realized by a message passing algorithm. Xiong *et al.* [11] design and train an inference procedure via a sequence of predictions from simple machine learning modules instead of using a single graphical model for modeling the contextual relations between points.

In this paper, we use MRF to model the contextual relations between points, but the way we obtain nodes and edge potentials of Markov networks are different from those in [2][3][4][1]. We apply standard machine learning algorithms, such as RF and SVM, for single point classification as a preprocessing step, and calculate node potentials from those classification results. We directly use point distance for edge potentials without learning edge weights. We find that RF is superior to max-margin style learning [2][1] because the randomized procedure in the training stage avoids overfitting, especially when the amount of training examples are huge and there is noise in the training data. In Section V-B, we show that RF consistently produces better classification results than those obtained from SVM, a typical max-margin style learning algorithm.

III. Simplified Markov Random Fields Using Machine Learning as a Preprocessing Step

In point cloud classification, we focus on Markov networks over discrete variables $\mathbf{Y} = \{Y_1, ..., Y_N\}$, where each variable corresponds to a point in the cloud that belongs to one of the K labels, i.e., $Y_i \in \{l_1, ..., l_K\}$. A Markov network for $\mathbf{Y}$ defines a joint distribution over $\{l_1, ..., l_K\}^N$, and such a distribution can also be defined by an undirected graph $(\mathscr{V}, \mathscr{E})$, where each node in $\mathscr{V}$ corresponds to a point in 3D and each edge in $\mathscr{E}$ corresponds to the physical proximity of the two points the edge links. In this paper, we use *pairwise* Markov networks [2], where nodes and edges are associated with potentials $\phi_i(Y_i)$ and $\phi_{ij}(Y_i, Y_j)$, respectively. Intuitively, $\phi_i(Y_i)$ encodes a point's individual preference for different labels, whereas $\phi_{ij}(Y_i, Y_j)$ encodes the interactions between labels of the two points the edge links. Once a Markov network is constructed with node and edge potentials, we can apply energy minimization algorithms [14] to find the best possible label assignment to all the points.

A. Preprocessing Using Machine Learning

A point's node potential, $\phi_i(Y_i)$, encodes its preference for different labels. In previous work [1][2][3][12], $\log \phi_i(Y_i)(l) = \mathbf{w}_n^l \cdot \mathbf{x}_i$, where $\mathbf{w}_n^l$ is the learned node weight vector for label l and $\mathbf{x}_i$ is the feature vector of the ith node. In this paper, we calculate $\log \phi_i(Y_i)$ from per-point classification results obtained from RF or SVM, two standard classifiers that we are most interested in.

1) Random Forest Classifier: Random Forest (RF) [15], also called random trees, has been successfully used for tasks such as human pose recognition [16] and image segmentation [17]. It runs efficiently on large amount of data with better classification accuracy than other algorithms because the randomized procedure in the training stage avoids over-fitting, especially when the amount of training examples are huge and there is noise in the training data.

Each tree T_t in the forest is trained from different training set Θ_t, which is a randomly selected subset of the entire training data Θ such that $|\Theta_t| = \alpha_t|\Theta|$, where $\alpha_t \in (0,1)$ and $|\cdot|$ denotes size. At each node N_s of T_t, a subset of feature variables ν_s is randomly selected from all the feature variables ν such that $|\nu_s| = \beta_s|\nu|$, where $\beta_s \in (0,1)$. From ν_s, we select a feature variable for split that produces best information gain calculated from entropy, and then split Θ_t into left and right subsets. We recursively perform random feature variable selection and node splitting based on information gain for left and right subtrees. The stopping criteria is that either T_t reaches maximum depth or the remaining examples have the same classification. The randomization of Θ_t for T_t makes RF robust to noise in the training dataset, and the randomization of ν_s at each splitting node eliminates the necessity of feature selection before the training stage. At the prediction stage, RF takes the feature vector of a point as the input, classifies it with each decision tree in the forest, and outputs the label l that received the majority votes or outputs $p(l)$, the distribution of votes (preference) for labels.

2) Support Vector Machines: SVM [18] is initially invented for building an optimal binary classifier. It maps feature vectors into a higher-dimensional space using kernel functions, such as linear function and radial basis function (RBF), and then it calculates an optimal hyper-plane in the space that best fits the training data. The solution is optimal because the margin between the separating hyper-plane and the nearest feature vectors from both classes is maximal, and it is one of the max-margin style learning algorithms. Feature vectors that are closest to the hyper-plane are called support vectors. Now, SVM has been extended for multi-class classification and used for point cloud classification in [9]. One drawback of SVM is that it is not a robust method. If there is noise in the training data, SVM tends to find the best separating hyper-plane that also fits noise. At the prediction stage, SVM takes the feature vector of a point as the input, maps it into the high-dimensional space, and outputs the predicted label l based on which side of the hyper-planes the feature vector falls on.

B. Markov Network Construction

We calculate node potentials $\phi_i(Y_i)$ based on the classification results obtained from point-wise classifier such as RF and SVM. For a node i, $\phi_i(Y_i) = p(l)$ if a classifier outputs a normalized distribution of preference for labels. Otherwise, $\phi_i(Y_i) = (\theta_1, ..., \theta_K)$ with $\theta_l = 1$ and $\theta_k = 0$, for $k \in [1,K]$ and $k \neq l$, if the classifier outputs a predicted label l. In previous work [1][2][3][12], edge potentials are calculated as $\log \phi_{ij}(Y_i, Y_j) = \mathbf{w}_e^l \cdot \mathbf{x}_{ij}$, where $\mathbf{w}_e^l$ is the learned edge weight vector for label l and $\mathbf{x}_{ij}$ is the feature vector of the edge ij. In this paper, we use Potts model that rewards pairs of points of the same label, and calculate edge potentials directly from point distances as follows:

$$\begin{aligned} \phi_{ij}(Y_i, Y_j) &= \omega_{ij}, \text{ if } Y_i = Y_j \\ &= 0, \text{ otherwise,} \end{aligned} \tag{1}$$

where ω_{ij} encodes the physical proximity between node i and j. We make an edge between nodes if their distance is smaller than r. Based on that, we detect false edges between nodes and break them using robust neighborhood filtering (see Section IV for details), in order to reduce over smoothed-out effect after minimizing energy for Markov networks.

C. Energy Minimization

Szeliski *et al.* [14] formulate maximum a posteriori estimation of a Markov network into an energy minimization problem, and the energy function E has two terms: data energy E_d and smoothness energy E_s. Data energy penalizes solutions that are inconsistent with the observed data, and smoothness energy enforces spatial coherence. Minimizing E_d and E_s is equivalent to maximizing node and edge potentials. In Eqn. (2), we re-formulate potentials into data and smoothness terms.

$$\begin{aligned} E &= E_d + \lambda E_s \\ &= \sum_{i \in \mathscr{V}} \sum_{l=1}^{K} (1 - \phi_i(Y_i)(l)) \\ &\quad + \lambda \sum_{ij \in \mathscr{E}} d_{ij}(1 - \delta(|Y_i - Y_j|)), \end{aligned} \tag{2}$$

where d_{ij} is the physical distance between node i and j, and δ is the unit impulse function. Graph cuts [19][20][21] and loopy belief propagation [22] are two popular optimization algorithms used to minimize an energy function in Eqn. (2).

IV. Robust Neighborhood Filtering

Geometric information contained in 3D point clouds is implicit and purely represented by the spatial arrangement of the observed points. Defining a local neighborhood, $\mathscr{N}_p$, around a 3D point p is essential for both geometric feature extraction and contextual relation construction in MRF. k-nearest neighbor (k-NN) and r-radius neighbor (r-RN) are two commonly used algorithms for finding $\mathscr{N}_p$. k-NN defines $\mathscr{N}_p$ as k closest points in distance to p, whereas r-RN includes all points that are within r distance to p as $\mathscr{N}_p$.

Finding k-NN of p is very fast using kd-tree data structure. However, geometric statistics extracted from $\mathscr{N}_p$ found by k-NN is largely dependent on the density of the point cloud. For a certain value of k, $\mathscr{N}_p$ can be under-representative for the dense part of a point cloud because the k points are too close to each other to form a spatial span for extracting valid local geometric features. Therefore, we use r-RN to find $\mathscr{N}_p$ in this paper, even though it is a little bit slower than k-NN.

Ideally, points in $\mathscr{N}_p$ should be of the same label as p. When finding neighbors using r-RN, it is inevitable to include points from other category (we call them outliers), especially at the intersections of different objects. Those outliers will

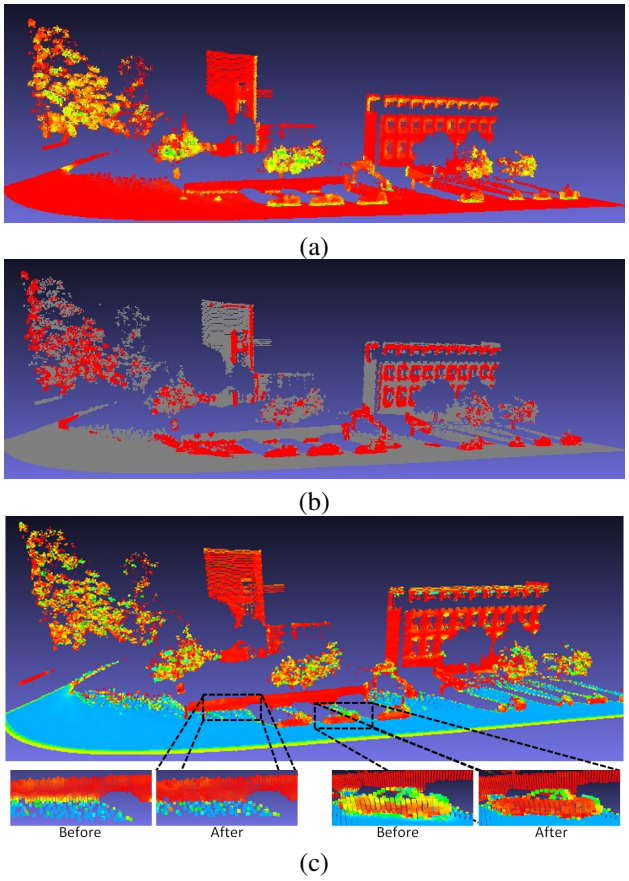


Fig. 2. Robust neighborhood filtering. Values of σ_p (a) and values of v_{nz} (c) are visualized in color with redder being smaller and bluer being larger. In (c), we also show two zoomed-in details of intersection areas before and after robust neighborhood filtering. All points that pass both point-ness thresholding and dominant plane criteria are highlighted in red in (b).

poison local geometric statistics when extracting features, and also lead to over smoothed-out effect after minimizing energy for Markov networks. At the training stage, we can easily remove outliers in $\mathcal{N}_p$ because ground truth labels are available. However, at the prediction and inference stages, we have to make extra effort to exclude outliers.

We locate points near intersections and exclude outliers in their neighborhood in two steps: 1) thresholding on point-ness value, and 2) fitting a dominant plane. Points near intersections tend to have large point-ness value, σ_p (defined in [12]), which is a measure of scatterness of a local neighborhood. Fig. 2a shows the value of σ_p visualized in color of a point cloud C, and we locate points whose σ_p is greater than a threshold, τ_p, calculated as follows:

$$\tau_p = m_p + \omega_p \mathrm{MAD}_p, \tag{3}$$

where m_p and MAD_p are the median and the MAD (Median Absolute Deviation) of σ_p over all points in C, respectively, and ω_p is the MAD multiplier that controls outlier tolerance. From Fig. 2a we see that porous objects such as vegetation also have large σ_p values. The scatterness of vegetation points is due to their scattering nature, whereas the scatterness of points near intersections is due to a mixture of geometric entities brought in by different objects. To differentiate these two types of scatterness, for every point that passes point-ness thresholding, we fit a dominant plane, P, for its $\mathcal{N}_p$ using RANSAC. If the number of inliers is more than a half of the total points in $\mathcal{N}_p$, P is the dominant plane and points that are not on P are considered as from different geometric entities, i.e. outliers, that should be excluded. Fig. 2b highlights in red all the points that pass both point-ness thresholding ($\sigma_p > \tau_p$) and dominant plane fitting criteria. Points near the intersection between vehicle and ground, and between low facade and ground are highlighted. A few facade points get highlighted due to their local bumpiness, and a few vegetation points are also highlighted. We discuss in Section V-C how they affect the final classification results.

Fig. 2c shows the value of v_{nz} for each point in C visualized in color after robust neighborhood filtering, and v_{nz} is the scalar projection of a point's normal direction, $\vec{v}_n$, onto the z-axis. We also present zoomed-in details around the intersections between facade and ground, and car and ground before and after robust neighborhood filtering. After neighborhood filtering, values of v_{nz} for facade points near the intersection with ground are more consistent with facade points higher from the ground. In addition, values of v_{nz} for car points are less noisy than those before filtering, and thus, they more clearly indicate that cars are nearly vertical objects. Section V-B presents more results showing that robust neighborhood filtering removes false edges in Markov networks and improves final classification results.

V. Experiments

A. Experiment Setup

We use the CMU Oakland 3D point cloud data set[2], which was collected from a moving platform in an urban environment. The dataset contains 17 different scenes, having over 1.6 million points and 44 semantic categories in total. Around 1.4 million points with five categories (vegetation, wire, pole/tree trunk, ground, and facade) are used in [1]. In this paper, we present results using the exact same data as used in [1] for comparison purposes. In separate experiments, we include vehicle points to demonstrate that our method is able to deal with scenes with more different objects. The six-category data contains around 1.5 million points.

We implement 6 geometric features, with 3 spectral features [12], 2 directional features, and 1 global contextual feature. Spectral features, $\{\sigma_p, \sigma_s, \sigma_l\}$, measure the point-ness, surface-ness, and linearity of the local neighborhood of a point. Directional features capture the local orientation, and their values are the scalar projection of the locally estimated tangent and normal directions, $\{\vec{v}_t, \vec{v}_n\}$, onto the z-axis. Spectral and directional features are calculated over r-RN with $r = 0.6$ m. When performing robust neighborhood filtering, we set $\omega_p = 10$ in Eqn. (3). The last feature is the height of a point with respect to the ground plane, the largest horizontal plane fitted over the entire points of a scene.

[2]Available at http://www.cs.cmu.edu/~vmr/datasets/oakland_3d/cvpr09/doc

The parameters for RF are set as follows: $\alpha_t = 2/3$, $\beta_s = 1/\sqrt{|v|}$ (refer to Section III-A.1 for meanings of the variables), and α_t is the same for each tree in the forest and β_s is the same for each node in each tree. We set the maximum depth of a tree to be 10 and the maximum number of trees in the forest to be 100. For SVM, we use RBF as the kernel function. When constructing Markov networks, we set the maximum number of neighbors of a point to be 20. If a point has more than 20 neighbors, we randomly select 20 from them. We observe that for some scenes in the Oakland data set, points in dense regions, e.g. ground or vehicle points, have thousand of neighbors within 0.6 m radius. It is not necessary to build edges for each pair of neighborhood points because doing so will blow out the memory with very little, if any, reward. When minimizing the energy of Markov networks, we use graph cuts with expansion algorithm [19][20][21], and the maximum number of iterations is set to be 3.

B. Results

We present results obtained from different methods, and they are M^{3}N from [1], SVM_MRF, RF_MRF, and RNF_RF_MRF. SVM_MRF and RF_MRF use classification results from SVM and RF, respectively, for calculating node potentials of Markov networks. RNF_RF_MRF performs robust neighborhood filtering as mentioned in Section IV for feature extraction at RF's prediction stage and for removing false edges when constructing Markov networks. All four methods perform label-based neighborhood filtering (remove points in $\mathcal{N}_p$ whose label is different from p) at their training stage to reduce noise in the training data.

We use precision, recall, and $F1$ scores to quantify and compare classification performance of different methods. The $F1$ score of a label l is the harmonic means of its precision P_l and recall R_l, and $F1 = 2P_lR_l/(P_l + R_l)$. The average of per-class $F1$ score is a better metric than overall accuracy because the latter can hide poor performance of classes with few samples, and we observe such a class imbalance in the Oakland data set we use. Besides precision, recall, and $F1$ scores, we compare the processing time of different methods because we are also concerned about real time implementation on mobile robotic systems.

1) Training on Single Scene: Table I shows the five-category classification results using an entire single scene, *Oak15*, for training, and *Oak15* is the same training data used in [1]. Among the six methods, M^{3}N HOC constructs Markov networks using both *pairwise* and high-order cliques, and the rest five methods only consider *pairwise* contextual relations. RNF_RF_MRF yields the best average $F1$ score among *pairwise* methods. There is no significant improvement after applying robust neighborhood filtering because there are not as many inter-label intersections in the five-category data as those in the six-category data, which we will show later.

The results of M^{3}N PAIR* in Table I are reproduced using the code provided along with the paper [1], and they should be close to the results of M^{3}N PAIR. However, we see some noticeable difference in recall for wire and facade points, and we speculate that it might be related to the following setup when we reproduce the results. First, we perform label-based neighborhood filtering when extracting features at the training stage, which might not be the case in [1]. Second, we set the number of iterations to be 3 when learning MRF weight vectors, and the number is not mentioned in [1]. There might be other subtle inconsistency in the experiment setup that we are not aware of.

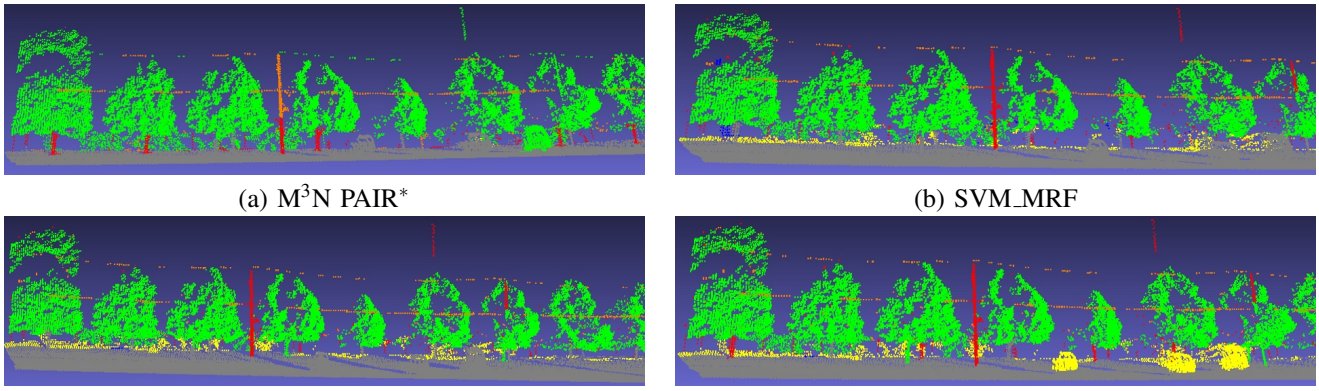
		veg.	wire	pole/ trunk	grd.	facade
P	M^{3}N PAIR [1]	0.98	0.26	0.18	**1.00**	0.90
	M^{3}N HOC [1]	**0.99**	**0.50**	0.26	**1.00**	0.91
	M^{3}N PAIR*	0.90	0.23	0.27	0.99	**0.95**
	SVM_MRF	0.97	0.25	0.25	0.99	0.94
	RF_MRF	0.97	0.48	0.33	**1.00**	0.79
	RNF_RF_MRF	**0.99**	0.38	**0.37**	**1.00**	0.78
R	M^{3}N PAIR [1]	0.87	0.89	**0.83**	0.99	0.88
	M^{3}N HOC [1]	0.93	**0.90**	0.81	**1.00**	0.88
	M^{3}N PAIR*	**0.94**	0.06	0.80	0.99	0.71
	SVM_MRF	0.92	0.65	0.59	0.99	0.87
	RF_MRF	0.89	0.80	0.57	0.99	0.91
	RNF_RF_MRF	0.88	0.83	0.70	0.99	**0.93**
$F1$	M^{3}N PAIR [1]	0.702				
	M^{3}N HOC [1]	**0.778**				
	M^{3}N PAIR*	0.645				
	SVM_MRF	0.709				
	RF_MRF	0.755				
	RNF_RF_MRF	0.757				

TABLE I

FIVE-CATEGORY CLASSIFICATION RESULTS USING AN ENTIRE SINGLE SCENE, *Oak15*, FOR TRAINING. P = PRECISION, R = RECALL, AND $F1$ = AVERAGE F1 SCORES OVER LABELS.

We add vehicle points as one more category because vehicles are common objects in urban scenes. Table II shows the six-category classification results using scene *Oak05* for training. We choose *Oak05* instead of *Oak15* because there are no vehicle points in the latter scene. RNF_RF_MRF yields the best average $F1$ score among the four methods. In particular, we see significant precision and recall improvement for vehicle points after applying robust neighborhood filtering (M^{3}N PAIR* has the highest precision 0.43, but considering its low recall 0.04, we do not think it performs better than RNF_RF_MRF). Fig. 3 shows example classification results in quality of a scene in the data set using the four methods. Without applying robust neighborhood filtering, nearly all vehicle points are classified as ground points in Fig. 3a, 3b, and 3c because of the over smoothed-out effect after energy minimization of Markov networks. In Fig. 3a, pole points are wrongly classified as wire points at the higher half of the pole, which does not happen for the other three methods. Similar to this issue, high facade points are wrongly classified as vegetation points in Fig. 1b. It seems that M^{3}N PAIR* sets a height threshold for facade and wire points in those scenes.

Comparing Table I and II, we see a drop of average $F1$ score for each method in Table II, and there are two reasons for that. First, *Oak15* contains more variety of data



(a) M^{3}N PAIR* (b) SVM_MRF

(c) RF_MRF (d) RNF_RF_MRF

Fig. 3. Comparison of six-category classification results on a sample scene using *Oak05* for training. Color scheme: green-vegetation, orange-wire, red-pole/trunk, gray-ground, blue-facade, yellow-vehicle

		veg.	wire	pole/ trunk	grd.	facade	veh.
P	M^{3}N PAIR*	0.61	0.53	**0.49**	0.96	**0.94**	**0.43**
	SVM_MRF	0.77	**0.59**	0.44	0.96	0.88	0.07
	RF_MRF	0.85	0.48	**0.49**	0.96	0.87	0.17
	RNF_RF_MRF	**0.92**	0.48	0.39	**1.00**	0.68	0.34
R	M^{3}N PAIR*	**0.98**	0.62	0.42	**0.99**	0.01	0.04
	SVM_MRF	0.94	**0.65**	0.61	0.98	0.57	0.03
	RF_MRF	0.94	0.58	0.53	0.98	0.65	0.13
	RNF_RF_MRF	0.92	0.61	**0.66**	0.98	**0.70**	**0.33**
*F*1	M^{3}N PAIR*	0.473					
	SVM_MRF	0.612					
	RF_MRF	0.631					
	RNF_RF_MRF	**0.660**					

TABLE II

SIX-CATEGORY CLASSIFICATION RESULTS USING AN ENTIRE SINGLE SCENE, *Oak05*, FOR TRAINING.

		veg.	wire	pole/ trunk	grd.	facade	veh.
P	SVM_MRF	0.73	**0.72**	0.56	0.97	0.13	0.43
	RF_MRF	0.95	0.70	0.57	0.98	**0.96**	0.71
	RNF_RF_MRF	**0.97**	0.67	**0.60**	**0.99**	0.93	**0.77**
R	SVM_MRF	0.91	**0.74**	0.52	0.99	0.00	0.69
	RF_MRF	**0.96**	0.66	0.55	0.99	0.88	0.64
	RNF_RF_MRF	0.93	0.69	**0.62**	**1.00**	**0.91**	**0.84**
*F*1	SVM_MRF	0.598					
	RF_MRF	0.794					
	RNF_RF_MRF	**0.825**					

TABLE III

SIX-CATEGORY CLASSIFICATION RESULTS USING 3% RANDOMLY SELECTED POINTS FROM EACH CATEGORY IN EACH SCENE FOR TRAINING.

for training than *Oak05*, except that the former does not have vehicle points. Facade points in *Oak05* all lie on one plane, whereas facade points in *Oak15* lie on several planes that have different normals. In addition, there are no low facade points in *Oak05*. Second, classifying more categories of object introduces more confusion. Low facade points and vehicle points are usually confused with each other because they are of the similar hight and they are both plenary. We could have picked other scenes for training that have more variety in facade points, but they might not have enough wire or pole points. It is difficult to select a single scene that contains good training examples for each category, and we find that training using *Oak05* produces relatively better results than training on other scenes.

2) Training on Randomly Selected Points: In order to obtain training examples of larger variety, we randomly select a certain percentage of points from each category in each scene, instead of training on an entire single scene. Table III shows the six-category classification results using 3% randomly selected points for training. RNF_RF_MRF yields the best average $F1$ score, and most noticeably, it significantly improves precision and recall for vehicle points. Comparing Table II and III, average $F1$ scores of RF_MRF and RNF_RF_MRF improve, especially for facade and vehicle points. However, the performance of SVM_MRF does not improve, and its gap to RF_MRF and RNF_RF_MRF increases. In Table III, SVM_MRF has very low precision and recall for facade points, and we find out that almost all of the facade points are classified as vegetation points by SVM_MRF. One explanation is that the bumpiness of facade points near window areas is confused with the scatterness of vegetation points. From another angle, it shows that SVM is not as robust as RF when dealing with noisy data.

Comparing Table I and III, we find that the average $F1$ scores improve for RF_MRF and RNF_RF_MRF in Table III, even though they deal with one more category. Having a more variety of training examples picked from each scene instead of just from an entire single scene improves the classification results for a robust classifier such as RF. We do not show M^{3}N results in Table III because it is not applicable when we randomly pick training points. M^{3}N has to construct a Markov network at the training stage in order to learn both node and edge weights, and isolatedly selected points cannot form a valid Markov network.

We also try other percentages of data used for training besides 3%, and Fig. 4 shows the average $F1$ score v.s. different percentage. Performance of RF_MRF and RNF_RF_MRF is

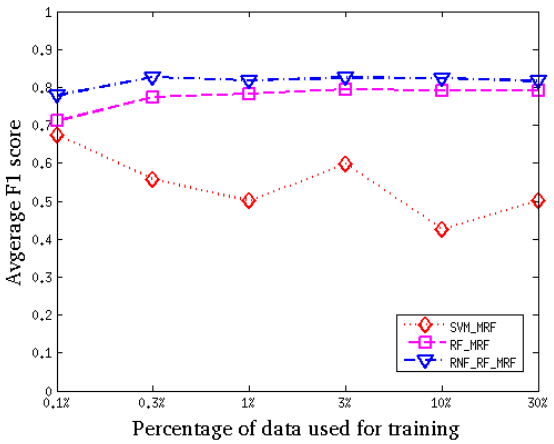


Fig. 4. Average $F1$ score v.s. different percentage of data used for training.

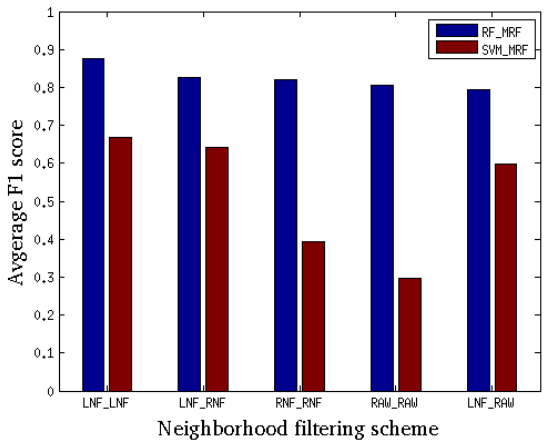


Fig. 5. Average $F1$ score v.s. different neighborhood filtering scheme when using 3% randomly selected points for training (see Section V-C for explanations of LNF, RNF, and RAW).

more stable than SVM_MRF because they are more robust to noise. Training on more data increases variety at the risk of introducing more noise. The ups and downs of SVM_MRF's $F1$ scores indicate that sometimes variety wins the battle and the other times noise wins. For RF_MRF and RNF_RF_MRF, training on 0.3% data produces similar results to those trained from 1% and 3% data.

3) Timing: In addition to classification accuracy, we are also concerned about the speed. Table IV shows the processing time per 10,000 points in seconds, broken into modules, when using *Oak05* for training. The machine we use is Intel(R) i7-2620 CPU @ 2.7GHz with 4GB memory. As mentioned at the beginning of Section V-B, we perform label-based neighborhood filtering at the training stage for all four methods. Therefore, time for extracting features used for training is the same for all four methods. When extracting features used for prediction, RNF_RF_MRF applies robust neighborhood filtering, and about 15% points need to go through such filtering, which takes about extra 1 second per 10,000 points. From Table IV, training using RF is much faster than SVM, even though their prediction time is about the same. M^{3}N PAIR* takes significantly longer time in the prediction and inference stages than the other three methods. Therefore, applying machine learning methods, such as RF and SVM, as a preprocessing step for calculating node potentials simplifies the work at the inference stage of MRF.

	feature		train	predict	infer	total
	train	predict				
M^{3}N PAIR*	11.87	11.89	39.00	11.58		74.34
SVM_MRF			48.06	0.22	1.11	73.15
RF_MRF			1.18	0.18	1.07	26.19
RNF_RF_MRF		12.96	1.18	0.18	0.90	27.09

TABLE IV
PROCESSING TIME PER 10,000 POINTS IN SECONDS, BROKEN INTO MODULES, WHEN USING *Oak05* FOR TRAINING.

C. Effect of Neighborhood Filtering

We investigate the effect of applying different neighborhood filtering schemes on the final classification results, and they are label-based neighborhood filtering (LNF), robust neighborhood filtering (RNF), and RAW. We mentioned LNF at the beginning of Section V-B, and explained RNF in detail in Section IV. RAW means taking every point within r radius of a point as its neighbors without applying any filtering. Fig. 5 shows the average $F1$ score v.s. different neighborhood filtering scheme when using 3% randomly selected points for training. We denote A_B as training using scheme A and predicting using scheme B, where A, B $\in$ (LNF, RNF, RAW). Results of LNF_LNF are shown in the figure just for reference, because using LNF in prediction is not possible as ground truth labels are not available in prediction. From the figure, we see that, generally, using RNF yields higher average $F1$ scores than using RAW. However, for RF_MRF, RAW_RAW produces better results than LNF_RAW. One explanation is that RF, at one hand, is robust to noise, and at the other hand, it learns subtle geometric structures berried in the raw neighborhood.

RNF does have some drawbacks. In Fig. 2b, a few facade points and vegetation points not on the intersection areas are highlighted. Then, in Table II and III, we see a drop of precision for facade points and a drop of recall for vegetation points after applying RNF, because some vehicle and vegetation points are classified as facade points. Removing outliers in the neighborhood also removes some detailed structure of vehicle and vegetation points which makes them look more like plenary facade points.

VI. CONCLUSIONS

This paper has presented an efficient classification method for semantic labeling of 3D point clouds. Our method uses simplified Markov networks to model the contextual relations between points. Instead of calculating node and edge potentials from learned weights, we obtain node potentials from

point-wise classification results using standard classifiers, and calculate edge potentials using physical distance between points. We also propose a novel robust neighborhood filtering for excluding outliers in the neighborhood, so that noise in local geometric statistics is reduced when we extract features and the number of false edges are reduce when we construct Markov networks. Our experimental results show good performance in terms of accuracy and speed when compared with state-of-the-art methods.

In the future, we would like to explore and use more contextual features besides the height to ground plane feature. For examples, besides ground, buildings, especially high buildings, are also importance geometric reference in the scene. Incorporating distances to building planes as additional features might reduce confusion between vehicle and facade points. We are also considering using supervoxels as the processing element, instead of individual points, for efficiency purposes. Moreover, using supervoxels opens up more areas for modeling high-order interactions between regions. Last but not least, we will experiment with data collected from a greater variety of 3D sensors.

REFERENCES

[1] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[2] 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 scan data," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, pp. 169–176.

[3] D. Munoz, N. Vandapel, and M. Hebert, "Directional associative markov network for 3-d point cloud classification," in *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, June 2008.

[4] E. H. Lim and D. Suter, "3d terrestrial lidar classifications with super-voxels and multi-scale conditional random fields," *Computer-aided Design*, vol. 41, pp. 701–710, 2009.

[5] H. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *Advances in Neural Information Processing Systems*, 2011.

[6] J. F. Lalonde, N. Vandapel, D. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *J. Field Robotics*, vol. 23, no. 10, pp. 839–861, 2006.

[7] R. Shapovalov, A. Velizhev, and O. Barinova, "Non-associative markov networks for 3d point cloud classification," in *Photogrammetric Computer Vision and Image Analysis*, vol. 38, September 2010, pp. 103–108.

[8] K. Lai and D. Fox, "3D laser scan classification using web data and domain adaptation," in *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[9] M. McDaniel, T. Nishihata, C. Brooks, and K. Iagnemma, "Ground plane identification using lidar in forested environments," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010.

[10] X. Xiong and D. Huber, "Using context to create semantic 3d models of indoor environments," in *Proceedings of the British Machine Vision Conference*, 2010, pp. 45.1–45.11.

[11] X. Xiong, D. Munoz, J. A. D. Bagnell, and M. Hebert, "3-d scene analysis via sequenced predictions over points and regions," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2011.

[12] D. Munoz, N. Vandapel, and M. Hebert, "Onboard contextual classification of 3-d point clouds with learned high-order markov random fields," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2009.

[13] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 1, pp. 433 – 449, May 1999.

[14] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, A. Agarwala, and C. Rother, "A comparative study of energy minimization methods for markov random fields," in *Proc. European Conf. Computer Vision*, 2006, pp. 16–29.

[15] L. Breiman and E. Schapire, "Random forests," in *Machine Learning*, 2001, pp. 5–32.

[16] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011, pp. 1297–1304.

[17] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[18] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[19] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, p. 2001, 2001.

[20] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, pp. 65–81, 2004.

[21] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, September 2004.

[22] M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. Proc. Int. Conf. Computer Vision, 2003.