Directional Associative Markov Network for 3-D Point Cloud Classification

Daniel Munoz, Nicolas Vandapel, and Martial Hebert
Carnegie Mellon University
{dmunoz,vandapel,hebert}@ri.cmu.edu

Abstract

In this paper we address the problem of automated three dimensional point cloud interpretation. This problem is important for various tasks from environment modeling to obstacle avoidance for autonomous robot navigation. In addition to locally extracted features, classifiers need to utilize contextual information in order to perform well. A popular approach to account for context is to utilize the Markov Random Field framework. One recent variant that has successfully been used for the problem considered is the Associative Markov Network (AMN). We extend the AMN model to learn directionality in the clique potentials, resulting in a new anisotropic model that can be efficiently learned using the subgradient method. We validate the proposed approach using data collected from different range sensors and show better performance against standard AMN and Support Vector Machine algorithms.

1. Introduction

Three-dimensional (3-D) point clouds are easier than ever to collect with the development of laser scanners and positioning systems. Accurate and dense point clouds can now be collected over large scale environments for numerous applications such as surveying, inspection, asset inventory, vegetation management, and autonomous scene understanding [7].

In this paper, we address the problem of automated interpretation of 3-D point clouds from scenes of urban and natural environments; our analysis is performed offline, from data acquired by a mobile mapping system. An example of our approach is illustrated in Figure 1 with five commonly found object classes[1]: ground, facade, scatter, pole/trunk, wire. We are interested in context-based 3-D point classification where, in addition to local features, a point's label is based on its neighboring points' label configuration. Markov Random Fields (MRFs) [8] constitute one of the options to account for neighboring information [14]. Such techniques proved to outperform classifiers based only on local features but tend to smooth out small components in the scene. To address this problem, we are interested in using a MRF variant called an Associative Markov Network (AMN) [11].

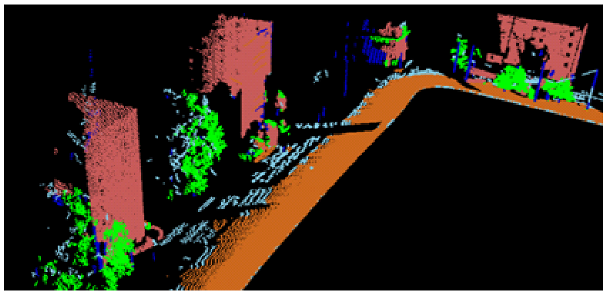


Figure 1. Urban environment classification with our approach.

AMNs and its variants in the literature [12, 1, 13] rely on local features and isotropic contextual information. With the isotropic model, the influence from surrounding points is only based on their label, regardless of their relative direction. We propose to extend the AMN to account for local directional information, thus producing an anisotropic model. The directional information can come from the relative position of the two points, or from a non-geometric feature, or from the local point topology. Our proposed approach is different, as we will show, from using local directional features. This natural extension is enabled by utilizing the recently proposed subgradient method shown to solve AMNs efficiently [10]. Originally, learning for AMNs was formulated as quadratic program which is very memory intensive when applied to 3-D point cloud processing; however, with the subgradient method, memory constraints are only linear in the amount of training data, thus allowing the development of a more expressive model. We compare the improvement in our model against the standard AMN and a linear Support Vector Machine (SVM) [5].

The paper is structured into five sections. In the next, various notations are introduced and background on the AMN and subgradient method is presented. The contributions of the paper follows in Section 3 and results in Section

[1]This paper is best viewed in color. Unless otherwise noted, the same color code labeling is used throughout the paper: brown for ground, red for facade, green for scatter, dark blue for pole/trunk, skye blue for wire.

4. Section 5 concludes the paper.

2. Associative Markov Network

2.1. Problem

Following the notation from [11], our classification task can be formalized as follows. Given a set of N random variables $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$, where each variable can obtain a value $Y_i \in \{1, \ldots, K\}$, find the assignment of values of $\mathbf{y} = \{y_1, \ldots, y_N\}$ to $\mathbf{Y}$ that maximizes some scoring function. In the context of 3-D point classification, each random variable represents a 3-D point and its value corresponds to the label it can be assigned. Formulating the classification task as a supervised learning problem, we want to learn a discriminative model that conditions the joint distribution on the features $\mathbf{x}$ that we can extract from the scene $P_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$, where $\mathbf{w}$ are the model parameters. The classification procedure is then broken into two steps: (1) learning the model parameters given labeled data $(\mathbf{x}, \hat{\mathbf{y}})$ and then (2) inferring the best assignments of a novel scene given its features.

2.2. Standard AMN formulation

A MRF, also called a Markov Network, defines a joint distribution for random variables $\mathbf{Y}$; it is represented as an undirected graph with N nodes for each random variable and edges $E = \{(i,j)\}|(i<j)$ that define the interactions between variables. Generally, a non-negative potential function is defined for cliques of arbitrary size in the graph; however, due to the requirement of efficient inference techniques, focus is generally on pairwise Markov Networks. This model only defines a node potential $\phi_i(y_i)$ for each node i and an edge potential $\phi_{ij}(y_i, y_j)$ for linked nodes i and j. These potentials measure the affinity[2] of the assignment to the variables in the cliques. A log-linear model is used to represent the dependence of the potentials on the features $\mathbf{x} = \{\mathbf{x_i}, \mathbf{x_{ij}}\}$ where $\mathbf{x_i} \in \mathbb{R}^{d_n}$ and $\mathbf{x_{ij}} \in \mathbb{R}^{d_e}$ are the features that describe node i and the relationship between nodes i and j, respectively. The log of the node potential is defined as $\log \phi_i(k) = \mathbf{w_n^k} \cdot \mathbf{x_i}$ where $k = y_i$ (the label value of node i) and $\mathbf{w_n^k} \in \mathbb{R}^{d_n}$ are the weights used when a node is assigned k.

Under the AMN framework, a variant of the Pott's model is used that penalizes differing assignments across an edge: $\forall k \neq l$, $\log \phi_{ij}(k,l) = \mathbf{w_e^{k,l}} \cdot \mathbf{x_{ij}} = 0$ and $\log \phi_{ij}(k,k) \geq 0$, where $\mathbf{w_e^{k,l}} \in \mathbb{R}^{d_e}$ are the weights used when linked nodes are assigned k and l. In order to ensure non-negativity in the edge potentials, the feature and weight vectors are constrained by $\mathbf{x_{ij}} \geq \mathbf{0}$ and $\mathbf{w_e^{k,k}} \geq \mathbf{0}$. Finally, changing the representation of an assignment $\mathbf{y}$ with a vector of $K \cdot N$ indicator variables where $\mathbf{y} = \{y_i^k, k, i | y_i^k = I(y_i = k)\}$, the log of the joint-conditional probability $\log P_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$ is given by:

$$\sum_{i=1}^{N} \sum_{k=1}^{K} (\mathbf{w_n^k} \cdot \mathbf{x}_i) y_i^k + \sum_{(ij)\in E} \sum_{k=1}^{K} (\mathbf{w_e^{k,k}} \cdot \mathbf{x}_{ij}) y_i^k y_j^k - \log Z_{\mathbf{w}}(\mathbf{x}) \quad (1)$$

where $Z_{\mathbf{w}}(\mathbf{x}) = \sum_{\mathbf{y}'} \prod_{i=1}^{N} \phi_i(y_i') \prod_{ij\in E} \phi_{ij}(y_i', y_j')$ is the partition function. Note although this value is intractable to compute, it does not depend on $\mathbf{y}$ which is essential for performing inference.

To abbreviate notation, define a $K(d_n + d_e)$ length row vector $\mathbf{w} = \{\mathbf{w_n}, \mathbf{w_e}\}$ with $\mathbf{w_n} = \{\mathbf{w_n^1}, \ldots, \mathbf{w_n^K}\}$ and $\mathbf{w_e} = \{\mathbf{w_e^1}, \ldots, \mathbf{w_e^K}\}$. Also redefine $\mathbf{y}$ to be a $K(N + |E|)$ column vector $\mathbf{y} = \{\mathbf{y_n}, \mathbf{y_e}\}^T$ with $\mathbf{y_n} = \{\ldots, y_i^1, \ldots, y_i^K, \ldots\}$ and $\mathbf{y_e} = \{\ldots, y_{ij}^1, \ldots, y_{ij}^K, \ldots\}$ where $y_{ij}^k = y_i^k \wedge y_j^k$. Finally, construct $\mathbf{X}$ to be a $K(d_n + d_e) \times K(N + |E|)$ matrix such that $\log P_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = \mathbf{wXy} - \log Z_{\mathbf{w}}(\mathbf{x})$. This matrix will contain the features repeated multiple times in the columns and padded with zeros appropriately.

Note that the inference task $\mathbf{y}^* = \arg\max_{\mathbf{y}} P_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \mathbf{wXy}$ is an integer program and is NP-hard. In [11], the authors show how to relax the integral constraints on $\mathbf{y}$, resulting in a linear program that finds the optimal solution when $K = 2$. For $K > 2$, a rounding procedure is performed that achieves an approximation. The authors also state that when $K = 2$, exact inference can be done by finding the min-cut of a specially constructed graph because the associative constraints on the negative edge potentials define a submodular[3] function [6]. For $K > 2$, performing an iterative min-cut algorithm, called α-expansion [3], also achieves an approximation. We refer to [11] and [3, 6] for more details.

Finding the optimal $\mathbf{w}$ is formulated as a max-margin learning problem. Given labeled data $(\mathbf{x}, \hat{\mathbf{y}})$, the goal is to find the weights that maximize the margin of confidence in $P_{\mathbf{w}}(\hat{\mathbf{y}}|\mathbf{x})$ versus $P_{\mathbf{w}}(\mathbf{y}|\mathbf{x})\ \forall \mathbf{y} \neq \hat{\mathbf{y}}$. This learning problem is formulated as the following convex program:

$$\begin{array}{ll} \min\limits_{\mathbf{w},\xi} & \frac{1}{2}\|\mathbf{w}\|^2 + \xi \\ \text{s.t} & \mathbf{wX\hat{y}} + \xi \geq \max\limits_{\mathbf{y}} \mathbf{wXy} + \mathcal{L}(\mathbf{y}) \end{array} \quad (2)$$

where ξ is a slack variable that represents the gap in the total energy between the optimal and achieved solutions and $\mathcal{L}(\mathbf{y})$ is a loss function which measures the error of classification. As in [11] and [1], we use the Hamming distance between the true and achieved assignments for our loss function. In [11], the authors show how to substitute the dual of the inference LP to bound the non-linear constraint which then results in a valid quadratic program and can then be solved by optimization software. Again, we refer to [11] for more details.

[2]The affinity value is also referred to as the energy of the clique.

[3]A function of two binary variables $E(\alpha, \beta)$ is submodular if and only if $E(0,0) + E(1,1) \leq E(0,1) + E(1,0)$

2.3. Subgradient method for learning

In [9, 10], the authors show that it is possible to solve Program 2 by writing the constraint in the objective function, due to the slacks being equal at the optimal condition, and then taking the subgradient of the resulting objective function. Thus, the AMN regularized cost function is:

$$c(\mathbf{w}) = \frac{\lambda\|\mathbf{w}\|^2}{2} + \max_{\mathbf{y}}(\mathbf{wXy} + \mathcal{L}(\mathbf{y})) - \mathbf{wX\hat{y}} \qquad (3)$$

The key to compute the subgradient of Equation 3 is to use the property: if $f(a,b)$ is differentiable in a, then $\nabla_a f(a,b^*)$ is a subgradient of the convex function $\max_b f(a,b)$ for $b^* \in \arg\max_b f(a,b)$. Therefore, a subgradient $g_\mathbf{w} \in \partial c(\mathbf{w})$ is:

$$g_\mathbf{w} = \lambda\mathbf{w} + \mathbf{Xy}^* - \mathbf{X\hat{y}}$$

As previously mentioned, solving $\max_\mathbf{y}(\mathbf{wXy} + \mathcal{L}(\mathbf{y}))$ can be done with graph cuts or an LP. Starting with $\mathbf{w} = \mathbf{0}$, the solution is then achieved through descent until convergence, or T iterations, using the update rule at time t:

$$\mathbf{w_{t+1}} = \mathcal{P}_\mathcal{W}[\mathbf{w_t} - \alpha g_{\mathbf{w_t}}]$$

where $\mathcal{P}_\mathcal{W}$ projects $\mathbf{w}$ onto a convex set $\mathcal{W}$ formed by any specific convex constraints on $\mathbf{w}$; for AMNs, this projection enforces any negative $\mathbf{w_e}$ to become 0. Typical step-sizes are $\alpha = \frac{c}{t}$ and $\alpha = \frac{c}{\sqrt{t}}$, for some positive c.

3. Directional Associative Markov Network

3.1. Motivation

Applications of AMNs for 3-D point cloud classification have proved to do well when classifying large, dominant structures in the scene such as vegetation, buildings or walls, and the ground plane [12, 1]. However, in most urban environments, there exist finer objects such as branches, posts, utility poles, and power-lines that are harder to perceive with laser scanners. In addition, these labels prove more challenging to classify when in the vicinity of data from more dominant labels, such as vegetation, because the AMN prefers to spatially maintain the same labels. Observe that Equation 1 is maximized when the labels of two nodes in an edge potential agree and the combination of the features and corresponding chosen weights is highest. Thus, when indicative features for the label cannot be computed, the label assignment is chosen to agree with its surroundings which may smooth away these small structures we are interested in.

3.2. Directionality

By accounting for directional information when computing our edge potentials we propose to address the limitations presented above. A basic way to accomplish this is to utilize the edge orientation when computing the energy. However, for 3-D point cloud processing the edge orientation is not expressive enough as the created edges will depend on the point density. Fortunately, most objects in the world often have an associated and well-defined direction that we can estimate. For example, tree trunks generally grow vertically, power-lines usually lie horizontally and we can estimate a local tangent vector at each point for both labels. Our goal is utilize this intrinsic information in our model so that a node's context accounts for its neighbors' local directions in addition to the labels. The idea behind this approach is to create a more expressive model that learns how to classify the data correctly when the estimated features, and consequentially the estimated local direction, are in a less separable or in a lower density region of the feature space. That is, we do not learn a single set of weights that tries to, overall, best model one class' features. Instead, we want to account for variation in feature estimation and learn multiple sets of weights for different locations in feature space that best model the class. By incorporating directional information in the AMN framework, we show how we can better preserve these smaller structures and improve the overall classification rate.

3.3. Anisotropic model

The standard AMN formulation is an isotropic model, that is, regardless of the orientation of the edge, the potentials are computed in the same manner. We propose using an anisotropic model where the weights chosen to compute the edge potentials depend on its label and defined direction; we call this new model a Directional AMN. We note that our approach extends to cliques of arbitrary size and is not limited to those of size two. The directional information is obtained by comparing a clique's *intrinsic* direction against a predefined *reference* direction when the clique is labeled k. The resulting angle between the intrinsic and reference directions is then binned. In addition to the label, the binned angle determines the sets of weights used to compute the clique potential, thus producing an anisotropic model. Figure 2 illustrates the following explanation of computing an anisotropic edge potential when its nodes are labeled k. For the two linked nodes (n_i, n_j) an intrinsic direction ($\vec{D}^l_{ij}$) is computed that describes the direction of the clique (edge) when its nodes are labeled k. This intrinsic direction can be defined arbitrarily. For example, the intrinsic direction could simply be the direction of the edge ($\vec{d_e}$), however, as previously mentioned, this would not provide much utility. Another example is to define a local feature direction for each node ($\vec{d_i}$) that describes the local direction when labeled k, such as the normal vector when fitting a plane, and then define the clique's intrinsic direction to be a function of each node's feature direction. The reference direction can be an absolute direction ($\vec{D}^A$), such as the vertical axis, or based on the local point cloud topology.

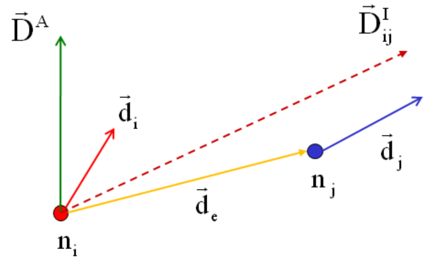
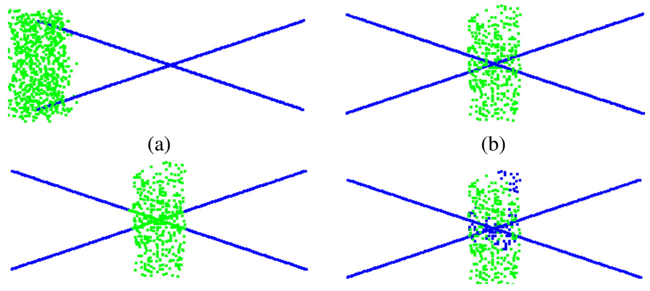


Figure 2. Directionality illustration.

It is important to note that the anisotropic model is different from an isotropic model with directional information in the features space; Figure 3 illustrates this claim. In this example, two artificial data sets were generated that contain two intersecting lines, parallel to the x-y plane, and are surrounded by randomly generated scattered points at two different locations. Note that this synthetic point cloud configuration mimics a common natural scene where power-lines are embedded in the vegetation. In the training set, illustrated in Figure 3-(a), the scattered points lie at the extremity of the lines, and for the testing set, illustrated in Figure 3-(b), the scattered points are moved to the intersection of the lines. In this example we use a standard and Directional AMN with the features defined in Section 4.4. Figure 3-(c), shows that the standard AMN smoothes out the classes we are interested in, while Figure 3-(d) shows that the Directional AMN performs a better job of preserving the small linear structure while increasing overall classification rate.



GT/Test	linear	scatter
linear	394	88
scatter	0	315

(e)

GT/Test	linear	scatter
linear	481	1
scatter	72	243

(f)

Figure 3. Difference between directional features and directional potentials, with the lines/scatter points in blue/green. (a) Training data. (b) Ground truth for the testing data. (c) Standard AMN. (d) Directional AMN. (e)/(f) Confusion matrix for standard/Directional AMN.

3.4. Directional AMN formulation

Incorporating the anisotropic potentials involves modifying the higher-order clique potentials from the original formulation, that is, modifying the edge potentials in the pairwise model. These clique potentials must now consider a direction term when computing the potential. For each label k, we parameterize a direction by binning the possible angle-space formed by the intrinsic direction against the reference direction when all nodes in the clique are labeled k. Remember that the intrinsic and reference directions are specific to each label. We denote the set of bins that constitute this space for label k as Θ_k. Note that the number of bins $|\Theta_k|$ for each label's angle-space are not necessarily equal. Therefore, the weight vector chosen when computing the clique potential is dependent on the clique's label k and the computed bin $\theta \in \Theta_k$ that the angle between the intrinsic and reference directions falls under, for label k. In the pairwise model, the anisotropic edge potentials are then defined $\log\phi_{ij}(k,k) = \mathbf{w_e^{k,\theta}} \cdot \mathbf{x_{ij}} \geq 0$ where $\theta \in \Theta_k$ is the computed bin, and $\forall k \neq l, \log\phi_{ij}(k,l) = 0$. Incorporating these changes, $\log P_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$ is proportional to:

$$\sum_{i=1}^{N}\sum_{k=1}^{K}(\mathbf{w_n^k}\cdot\mathbf{x}_i)y_i^k + \sum_{(ij)\in E}\sum_{k=1}^{K}\sum_{\theta\in\Theta_k}(\mathbf{w_e^{k,\theta}}\cdot\mathbf{x}_{ij})y_i^k y_j^k \Omega_{ij,k}^{\theta} \quad (4)$$

where $\Omega_{ij,k}^{\theta}$ is an indicator function defined to be one if the nodes in edge/clique (ij) are both labeled k and the angle between its intrinsic and reference direction lies in bin $\theta \in \Theta_k$.

As done with the standard AMN, we can relax Equation 4 into a linear combination. First, the set of edge weights $\mathbf{w_e}$ is redefined to be a $K \cdot d_e \cdot \sum_{k=1}^{K}|\Theta_k| = H$ length vector: $\mathbf{w_e} = \{\ldots, \mathbf{w_e^{k,1}}, \ldots, \mathbf{w_e^{k,|\Theta_k|}}, \ldots\}$. Intuitively, we would like to introduce indicator variables $y_{ij}^{k,\theta} = y_i^k \wedge y_j^k \wedge \Omega_{ij,k}^{\theta}$ to represent when the clique is labeled k and the extracted bin when labeled k is θ. However, this would require further constraints to ensure only one variable is "on" for each label and to prevent impossible extracted bins[4]. Instead, by defining $\mathbf{x}_{ij}$ to be a zero vector for impossible extracted bins, we can represent Equation 4 as $\mathbf{wXy}$ by redefining $\mathbf{X}$ to be a $H \times K(N + |E|)$ matrix. The new model can be solved using the subgradient method as before. Inference is easily performed through the min-cut framework with the α-expansion algorithm [3, 2, 6]. At each expansion step, we compute the potential of each clique. If all the nodes' labels in a clique agree, then the associated intrinsic and reference directions are determined for that clique and label. Using the resulting computed bin and label, the appropriate set of weights are then selected to compute the potential.

This new anisotropic model is related to, though significantly different from, [4] where the authors use an

[4]For a clique and label combination, only 1 bin can be computed.

anisotropic MRF. Their edge potential is dependent on the angle between the principal direction of the structure tensor and the edge defined by two nodes. No parameters are learned in their model. In contrast, we use the directionality to define an indicator function on how to compute the potential. We also learn multi-dimensional weights associated with the binned directions and the labels.

4. Experiments

4.1. Data sets and features

The results presented below were obtained using data collected from two Sick-based laser sensors. The first data set, coined the "sweeping" data set, was acquired using a sweeping Sick scanner from various static locations. A 400 x 800 range image is produced, with centimeter range resolution, and quarter degree angular resolution. The second data set, coined the "push-broom" data set, was produced using a static Sick laser mounted on a moving platform equipped with a navigation system. The vehicle drove in an urban environment at up to 20 km/h. The maximum range for the sensor for both data sets is approximately 60 m for vertical targets and 20 m along the ground with a sharp variation in point density.

The various data sets were hand labeled systematically into more than fifty different classes (See Figure 4). Labels were filtered out or collapsed into one of five labels (wire, pole/trunk, scatter, ground and facade). A total of half million 3-D points were labeled and used to produce results with ground truth for this paper. A total of more than five millions 3-D points corresponding to more than two kilometers traversed were classified and analyzed.

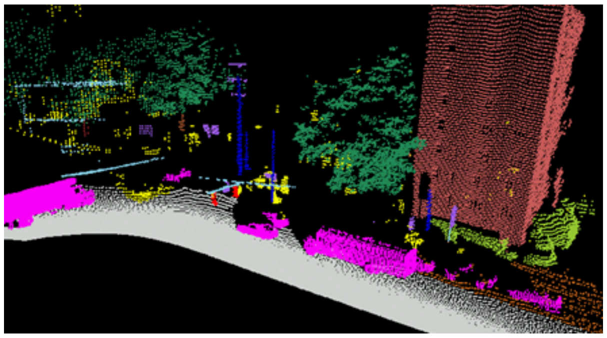


Figure 4. "Push-broom" data set. Example of ground truth data with vehicle (pink), pole (dark blue), wires (skye blue), paved road (grey), load bearing but not paved road (brown), buildings (dark red), bush (light green), and canopy (dark green).

We implemented three geometric features commonly used in spectral analysis of point clouds. We define $\lambda_2 \geq \lambda_1 \geq \lambda_0$ to be the eigenvalues of the scatter matrix M defined over a local neighborhood $\mathcal{N}_p$ around point p. These features capture the {point, surface, linear}-"ness" of the local geometry: $\{\sigma_p = \lambda_0, \sigma_s = \lambda_1 - \lambda_0, \sigma_l = \lambda_2 - \lambda_1\}$, respectively. We will refer to these as the spectral features. Next, we estimate the local tangent $\vec{v}_t$ and normal $\vec{v}_n$ vectors for each point by using the principal and least principal eigenvectors of M, respectively. We then compute the cosine and sine of the angles formed between the directions of $\vec{v}_t$ and $\vec{v}_n$ against the vertical and horizontal plane, resulting in four values. Though, depending on the local neighborhood, the estimated directions may be arbitrary. We estimate a confidence by scaling the values when using $\{\vec{v}_t, \vec{v}_n\}$ by $\{\sigma_l, \sigma_s\}/\max(\sigma_l, \sigma_p, \sigma_s)$, respectively. We will refer to these scaled values as the directional features. The actual node and edge features used for each experiment will be defined in their upcoming and respective subsection.

4.2. Model parameters and timing

Optimal parameters were obtained by maximizing the classification rate of various labeled data sets. For results reported on both data sets, we obtained the subgradient parameters $\lambda = 0.005$ and $\alpha = \frac{1}{2t}$. For the "sweeping" data, $T = 500$ and for the "push-broom" data, $T = 800$. The $\mathcal{N}_p$ was defined with a radius of 0.4 m for the "sweeping" data and 0.6 m for the "push-broom" data; we disregard points where $|\mathcal{N}_p| < 4$.

Results were computed on a Intel(R)-based 2.40 GHz processor with 4 GB RAM. We present timing analysis on the "push-broom" data set. The training set consisted of a graph with 18 898 nodes and 55 507 edges. Training took 151 minutes for the Directional AMN versus 148 minutes for the standard AMN. The ground truth testing set consisted of a graph with 385 611 nodes and 1 077 968 edges; 4 690 points were disregarded due to neighborhood size. On the test dataset, feature computation and graph construction completed in under 6.5 minutes, combined. Inference for the Directional AMN required 9.3 minutes versus 9 minutes for the standard AMN.

We constructed the graphs by iterating over the nodes and linking each node to its five nearest neighbors. We observed that the facade had the least amount of interactions with the other labels while scatter had the most. On the "push-broom" test set, the data was distributed as follows among the labels: 19.76% for scatter, 0.76% for wire, 0.67% for pole/trunk, 68.15% for ground, and 10.62% for facade.

4.3. Classifying the "sweeping" data set

In the first result, we collapsed the labels into three geometric labels: linear, surface, scatter. For the standard AMN, we found using only the spectral features as the node features and the concatenation of linked node features for the edge features gave the better results than when using the directional features. We also added a bias term for both features. For the Directional AMN we used the same features

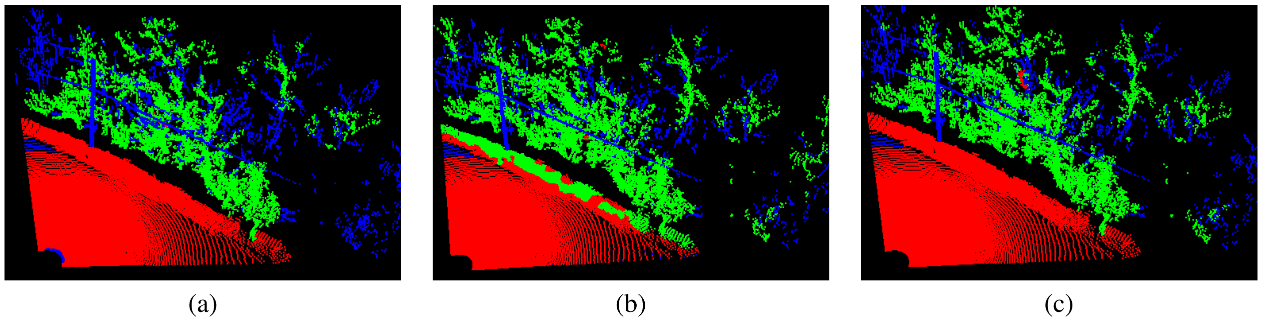


Figure 5. Classification results on "sweeping" data set for three labels (linear/scatter/surface). (a) AMN classification. (b) Directional AMN (linear, surface) classification. (c) Directional AMN (linear) classification.

	Linear SVM			Standard AMN			Dir. AMN (linear)			Dir. AMN (linear, surface)		
	linear	surface	scatter	linear	surface	scatter	linear	surface	scatter	linear	surface	scatter
Recall	0.797	0.997	0.080	0.948	0.966	0.721	0.892	0.995	0.816	0.876	0.934	0.839
Precision	0.172	0.794	0.986	0.187	0.998	0.985	0.295	0.997	0.981	0.312	0.999	0.838

Table 1. "Sweeping" data set. Confusion matrices with three classes and three features.

but experimented with two different anisotropic potentials for the linear and surface labels. For both the linear and surface potentials, we bin the angle between $\vec{v_t}$ and the horizontal plane, and the angle between $\vec{v_n}$ and the vertical into bins $\{[0,\pi/6],(\pi/6,\pi/2]\}$. We suffix in parenthesis which labels have anisotropic potentials.

In Table 1 we present recall and precision for classification results of a given scene using four different approaches: linear SVM, standard AMN, Directional AMN (linear), and Directional AMN (linear, surface). The overall error rates are 27.13, 10.48, 5.82, and 9.42 percent, respectively. Note the increase in the overall classification when using the Directional AMN (linear) versus the other classifiers. The difference between Directional (linear) and (linear, surface) AMN is most likely due to the training data containing no rough and angled terrain as seen under the pole in the testing data. The visual[5] classification results for only three of the classifiers are in Figure 5 due to the AMNs performing significantly better than the SVM.

In the next result, the linear class is separated into a wire (skye blue) and a pole/trunk (dark blue) class with the remaining linear points filtered out, such as thin branches. Anisotropic potentials are defined for both the wire and pole/trunk label to bin the space between $\vec{v_t}$ and the horizontal plane and vertical, respectively, into bins $\{[0,\pi/6],(\pi/6,\pi/2]\}$. We illustrate that with the same training set and only spectral features, we can perform classification of four classes due to directionality. Results with the Directional AMN (wire, pole/trunk) are presented in Figure 6 and the confusion matrix is presented in Table 2.

[5]Color code: red = surface, blue = linear, green = scatter

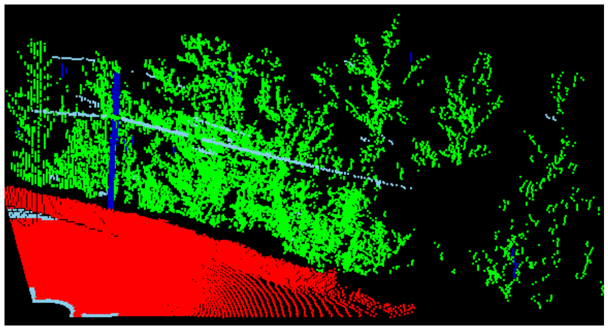


Figure 6. Four label classification on "sweeping" data using Directional AMN (wire, pole/trunk) with three geometric features.

Directional AMN (wire, pole/trunk) Error rate: 10.5%					
	wire	pole	ground	scatter	Recall
wire	446	0	0	46	0.906
pole	0	407	6	36	0.906
ground	2662	0	34087	5	0.927
scatter	196	79	5	15185	0.981
Precision	0.134	0.837	0.999	0.994	

Table 2. "Sweeping" data set. Confusion matrix with four classes and three features.

4.4. Classifying the "push-broom" data set

For the "push-broom" data we used five labels: ground, pole/trunk, wire, scatter, and facade. We compare Directional AMN (facade, pole/trunk, wire) against the standard AMN, where facade binned the angles between $\vec{v_n}$ and the horizontal plane into bins $\{[0,\pi/6],(\pi/6,\pi/2]\}$ and pole/trunk and wire use the same bins from the "sweeping" data experiment. For these results we found using the directional features in both models increased performance, and we note that the Directional AMN performed better in both. For the edge features, we concatenate two linked nodes' spectral features and compute a similarity feature for the

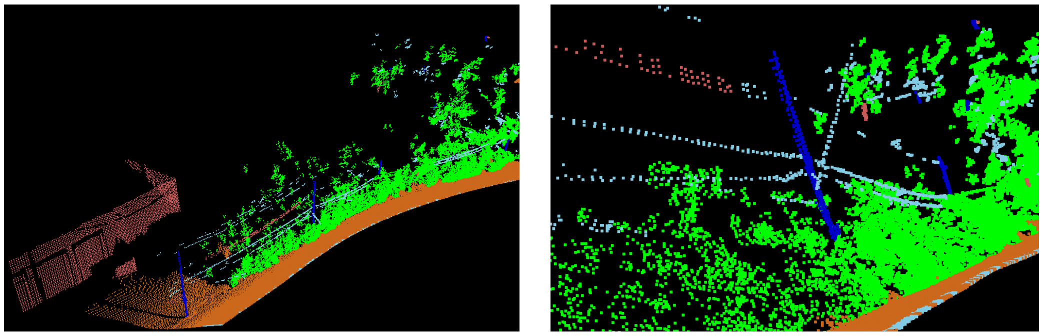


Figure 7. "Push-broom" data set. Classification of part of the ground truth subset into five labels. Right: Close-up view of pole and wire junction correctly classified.

directional features. This similarity feature is defined to be $1/(1+|df_i - df_j|)$ where df_i is a directional feature of node i.

Figure 7 shows results on part of the section used for quantitative performance evaluation. Note the close-up view of the pole and wires correctly labeled. Points not belonging to the five labels used for this evaluation were filtered out from the fully labeled ground truth data. We chose this approach to be able to correctly compare the classification results of the standard and Directional AMN with different features.

Table 3 presents the confusion matrix, recall and precision, for the Directional AMN computed over the subset with ground truth, over 390 000 points. The precision and recall for the standard AMN are also provided for comparison. As shown the Directional AMN is producing better precision and recall than the standard AMN for all labels. The most common error in classification is due to point density variation. This is clear with the precision of the wire and pole/trunk labels. Sections of ground far from the sensor tend to be mislabeled as wire. Low density coupled with occlusions, generate facade points being mislabeled as pole/trunk. The second common source of error is the inability of the features to capture the scene. For example bundle wires are misclassified as facade in Figure 7. Note that the quantitative performance evaluation presented here is on a much larger data set than usually reported in the literature. We processed the non-ground truth subsection of the "push-broom" data set, over 4.5 millions 3-D points. In such a case, all scene elements from the raw data are present. We present results for the best classifier, the Directional AMN (facade, pole/trunk, wire); qualitatively the classifier performs well, as shown in Figure 8. Objects not part of the training data, such as traffic lights and their support post are actually assigned to the closest geometrical label, respectively facade and linear. This is illustrated in Figure 9.

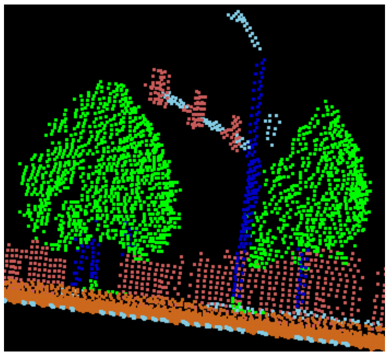


Figure 9. "Push-broom" data set. Classification on raw data.

5. Conclusions

In this paper we present a contribution to the problem of automated 3-D point cloud classification for scene interpretation. We extend the standard Associative Markov Network model to account for directional information, thus producing a new anisotropic model capable of representing accurately more complex scene structures than before. Recent developments in optimization with the subgradient method have allowed us to develop and learn this more complex model. We show how the proposed Directional AMN is different from using directional features with the standard AMN formulation. The approach is validated using several scans from a static ground scanner and using data accumulated by a push-broom sensor on a mobile platform. We produced performance evaluations on a very large manually labeled set, over four hundred thousand points in total. Qualitative evaluation was presented over the remaining data of over five millions 3-D points. The classification rate of the proposed approach was compared advantageously against SVM and the standard AMN, specifically for junctions and small components.

The limitations of the anisotropic model are restricted to classes where meaningful intrinsic and reference directions can be defined. For classes, such as scatter, where there exists no dominant direction, it is unclear how to define an

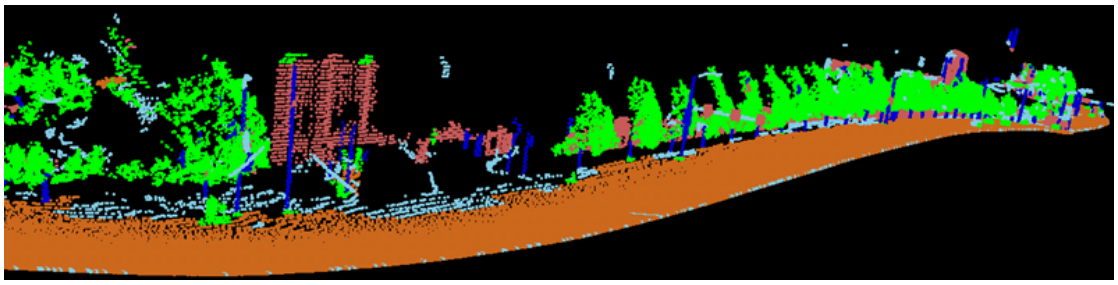


Figure 8. ”Push-broom” data set. Classification on raw data.

Dir. AMN	scatter	wire	pole/trunk	load bearing	facade	Recall	*Std. AMN Recall*
scatter	67187	3623	617	2059	2748	0.881	*0.856*
wire	145	2334	26	0	454	0.789	*0.778*
pole/trunk	92	38	2423	9	54	0.926	*0.899*
load bearing	1129	12342	0	249337	21	0.949	*0.945*
facade	440	394	5377	2570	32192	0.786	*0.672*
Precision	0.974	0.125	0.287	0.982	0.908		
Std. AMN Precision	*0.973*	*0.124*	*0.230*	*0.963*	*0.865*		

Table 3. ”Push-broom” data set. Confusion matrix for classification with five labels on 390 000 labeled points. The overall classification rate is 91.66% versus 89.67% for the standard AMN on the same features.

intrinsic direction that is representative of the class. Furthermore, this approach does not address typical errors due to point density that will disable the ability to accurately estimate local features and directions. Utilizing work from scale-space theory for feature estimation would be beneficial in our approach. Finally, constructing the graph in a smarter way to require fewer edges will limit inference time and step towards a real-time implementation; we are currently exploring alternative methods.

Acknowledgements

Prepared through collaborative participation in the Robotics consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-209912. The first author is supported through the Sandia National Laboratories Masters Fellowship Program. The authors would like to thank N. Ratliff and J. Bagnell for their discussions on the subgradient method.

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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9), 2004.

[3] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12), 2001.

[4] V. Grau, J. C. Downs, and C. Burgoyne. Segmentation of trabeculated structures using an anisotropic markov random field: application to the study of the optic nerve head in glaucoma. *IEEE Transactions on Medical Imaging*, 25(3), 2006.

[5] T. Joachims. *Advances in Kernel Methods - Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press, 1999.

[6] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.

[7] J.-F. Lalonde, N. Vandapel, and M. Hebert. Data structures for efficient dynamic processing in 3-d. *The International Journal of Robotics Research*, 26(8):777–796, August 2007.

[8] S. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 1995.

[9] N. Ratliff, J. Bagnell, and M. Zinkevich. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, 2006.

[10] N. Ratliff, J. Bagnell, and M. Zinkevich. (online) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, 2007.

[11] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *International Conference on Machine Learning*, 2004.

[12] R. Triebel, K. Kersting, and W. Burgard. Robust 3d scan point classification using associative markov networks. In *IEEE International Conference on Robotics and Automation*, 2006.

[13] R. Triebel, R. Schmidt, O. M. Mozos, and W. Burgard. Instance-based amn classification for improved object recognition in 2d and 3d laser range data. In *International Joint Conference on Artificial Intelligence*, 2007.

[14] C. Wellington, A. Courville, and A. Stentz. A generative model of terrain for autonomous navigation in vegetation. *International Journal of Robotics Research*, 25(12), 2006.