Airborne LiDAR Points Classification Based on Tensor Sparse Representation

N. Li[a,b,*], N. Pfeifer[b], C. Liu[a]

[a] College of Survey and Geoinformation, Tongji University, 200092, Shanghai, China - liuchun@tongji.edu.cn
[b] Deptartment of Geodesy and Geoinformation, Technische Universität Wien, 1040 Vienna, Austria - Norbert.Pfeifer@geo.tuwien.ac.at ; nan.li@tuwien.ac.at

ISPRS WG II/3

KEY WORDS: Point cloud, Sparse coding, Dictionary Learning

ABSTRACT:

The common statistical methods for supervised classification usually require a large amount of training data to achieve reasonable results, which is time consuming and inefficient. This paper proposes a tensor sparse representation classification (SRC) method for airborne LiDAR points. The LiDAR points are represented as tensors to keep attributes in its spatial space. Then only a few of training data is used for dictionary learning, and the sparse tensor is calculated based on tensor OMP algorithm. The point label is determined by the minimal reconstruction residuals. Experiments are carried out on real LiDAR points whose result shows that objects can be distinguished by this algorithm successfully.

1. INTRODUCTION

LiDAR point cloud classification in urban areas has always been an essential and challenging task. Due to the complexity in urban scenes, it is difficult to label objects correctly using only single or multi thresholds. Thus, research mainly focus on the use of statistical method for supervised classification of LiDAR points in recent years. Common machine learning methods include support vector machine (SVM) algorithm, adaboost, decision trees, random forest and other classifiers. SVM seeks out the optimal hyperplane that efficiently separates the classes, and the Gaussion kernel function can be used to map non-linear decision boundaries to higher dimensions where they are linear (Secord and Zakhor, 2007). Adaboost is a binary algorithm, but several extensions are explored for multiclass categorization, hypothesis generation routines are used to classify terrain and non-terrain area (Lodha et al., 2007). Decision trees can be used to carry out the classification by training data and make a hierarchical binary tree model, new objects can be classified based on previous knowledge (Garcia-Gutierreza et al, 2009) (Niemeyer et al., 2013). Random Forest is an ensemble learning method that uses a group of decision trees, provides measures of feature importance for each class (Guo et al., 2011) (Niemeyer et al., 2013), and runs efficiently on large datasets.

Those approaches barely consider the spatial distribution of points, which is an important cue for the classification in complex urban scenes. Some studies have applied graphical models to incorporate spatial context information in the classification. Graphical models take neighboring points into account, which allow us to encode the spatial and semantic

relationships between objects via a set of edges between nodes in a graph (Najafi et al. 2014). Markov network and conditional random field (CRF) are two mainstream methods to define the graphical model. However, a large amount of training data is necessary to obtain the classifier in those statistical studies. Anguelov et al (Anguelov et al. 2005) use Associated Markov Network (AMN) to classify objects on the ground. The study takes 1/6 points as training data and achieve an overall classification accuracy as 93%. Niemeyer et al (Niemeyer et al. 2014) use 3000 points per class to train the CRF model. Seven classes (grass land, road, tree, low vegetation, buildings gable, building flat, facade) are distinguished based on the CRF and the overall accuracy is 83%, which is a fine result in complex urban scene. As for statistical methods, Im (Im et al. 2008) uses 316 training samples (1%) to generate decision trees with an overall accuracy of 92.5%. Niemeyer (Niemeyer et al., 2013) uses 3000 samples per class to build random forest model, the overall accuracy achieves 83.7%. Moreover, Lodha uses half of dataset as training data through adaboost algorithm and the average accuracy is 92%. As a consequence, classifier training would be very time-consuming, especially when Markov network or CRF are used as classifier.

This paper aims to use as few training data as possible to achieve effective classification. Therefore, sparse representation-based classification is used in this paper. Sparse representation-based classification (SRC) is a well-known technique to represent data sparsely on the basis of a fixed dictionary or learned dictionary. It classifies unknown data based on the reconstruction criteria. SRC has been successfully applied to the processing of signals (Huang and Aviyente 2006) and images (Wright et al. 2009). Normally, the dimensional data has to be embedded into vectors in traditional methods. However, the vectorization breaks the original multidimensional structure of the signal and reduces the reliability of post processing. Therefore, some research formulates high dimensional data SRC problem in terms of tensors. Tensor extensions of the dictionary learning and sparse coding algorithms have been developed, such as Tensor MOD and KSVD for dictionary learning (Roemer et al. 2014), tensor OMP (Caiafa and Cichocki 2012). Moreover, tensor based representation has yielded good performance in high-dimensional data classification (Renard and Bourennane 2009), face recognition(Lee et al. 2015) and image reduction (Peng et al. 2014).

We represent LiDAR point as a 4-order tensor to keep feature description in their original geometrical space. With few training data, the dictionary is learned based on the Tucker decomposition(Kolda and Bader 2009). Then, the sparse representation of each point can be obtained by projecting the tensor onto dictionaries, which is expressed as a sparse tensor whose nonzero entries correspond to the selected training samples. Thus, the sparse tensors of points that belong to the same class should have similar structure. At last, the label of unknown points can be predicted by the minimum reconstruction residual from sparse tensor and dictionaries.

2. LIDAR CLASSIFICATION BASED ON SPARSE REPRESENTATION

2.1 Sparse Representation Classification

The sparsity algorithm is to find the best representative of a test sample by sparse linear combination of training samples from a dictionary (Wright et al. 2009). Given a certain number of training samples from each class, the sub-dictionary D_i from i^{th} class is learned. Assume that there are c classes of subjects, and let $D = ([D_1], [D_2], [D_3] \cdots [D_c])$ which is the overall structured dictionary over the entire dataset. Denote by y a test sample, the sparse coefficient x is calculated by projecting y on dictionary D, which is called sparse coding procedure.

SRC use the reconstruction error e_i associated with each class to do data classification. x_i is the sparse coefficient associated with class i, e_i is the reconstruction error from sub-dictionary in i class. The class label of y is then determined as the one with minimal residual.

$$e_i = \|y - D_i x_i\|_2 \quad class\ i = [1,2,\dots,c] \qquad (1)$$
$$\text{identify(y)} = \arg\min_i \{e_i\}$$

2.2 Tensor Representation of Lidar Points

2.2.1 Preliminaries on tensors

The tensor is to denote a multidimensional object, whose the elements are to be addressed by more than two indices. The order of a tensor, also known as modes (Kolda and Bader 2009), is the number of dimensions. Tensors are denoted as boldface italic capital letters, e.g., $\boldsymbol{A}$; matrices are denoted as italic capital letters, e.g., A; vectors are denoted as italic lowercase letters, e.g., a.

The tensor can be transformed into a vector or matrix, and this processing is known as unfolding or flattening. Given an N-order tensor $\boldsymbol{T} \in \mathbf{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \cdots \times \mathrm{I}_\mathrm{N}}$, the *n-mode* unfolding vector of tensor $\boldsymbol{T}$ is obtained by fixing every index except the one in the mode n (Yang et al., 2015). The n-mode unfolding matrix is defined by arranging all the n-mode vectors as columns of a matrix, i.e., the n-mode unfolding matrix $T_{(n)} \in \mathbf{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \cdots \mathrm{I}_{\mathrm{n}-1} \times \mathrm{I}_{\mathrm{n}+1} \cdots \times \mathrm{I}_\mathrm{N}, \mathrm{I}_\mathrm{n}}$. The n-mode product of a tensor $\boldsymbol{T} \in \mathbf{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \cdots \times \mathrm{I}_\mathrm{N}}$ with a matrix $U \in \mathbf{R}^{\mathrm{J} \times \mathrm{I}_\mathrm{n}}$ is denoted by $\boldsymbol{T} \times_{\boldsymbol{n}} \mathbf{U}$, and the processing can convert to product that each mode-n vector is multiplied to the matrix $\mathbf{U}$. so it can also be expressed in terms of unfold tensors:

$$\boldsymbol{Y} = \boldsymbol{T} \times_{\boldsymbol{n}} \mathbf{U} \Leftrightarrow Y_n = \mathbf{U}$$

The tucker decomposition is a form of higher-order principal component analysis. It decomposes a tensor $\boldsymbol{T} \in \mathbf{R}^{\mathrm{I}_1 \times \mathrm{I}_2 \times \cdots \times \mathrm{I}_\mathrm{N}}$ into a core tensor $\boldsymbol{C} \in \mathbf{R}^{\mathrm{J}_1 \times \mathrm{J}_2 \times \cdots \times \mathrm{J}_\mathrm{N}}$ multiplied by the matrix $\mathbf{U_1}, \mathbf{U_2}, \cdots \mathbf{U_N}$ along each mode. The matrix can be considered as the principal components in each mode. Since the tensor is generated according to the point spatial distribution, the principal components in attribute mode are spatial connection considered.

$$\boldsymbol{T} = \boldsymbol{C} \times_{\mathbf{1}} \mathbf{U_1} \times_{\mathbf{2}} \mathbf{U_2} \times \cdots \times_{\boldsymbol{n}} \mathbf{U_N}$$

2.2.2 Tensor representation of Lidar points voxel

In order to preserve spatial structure and attribution information, LiDAR points are represented as tensor data. In previous work, the LiDAR data is rasterized into feature images. The LiDAR tensor is generated by stacking images into 3-order tensor(Li et al., 2016). This paper consider each point voxel as a tensor. First of all, multiple attributes from raw LiDAR data are extracted to form a vector on the point $\mathbf{p}$, then the 3D neighborhood of the point $\mathbf{p}$ is selected as the voxel of point $\mathbf{p}$. After that, the voxel is represented as a 4-order tensor $\boldsymbol{T} \in \mathbf{R}^{\mathrm{I_x} \times \mathrm{I_y} \times \mathrm{I_z} \times \mathrm{I_a}}$ of point $\mathbf{p}$, where $\mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z}, \mathrm{I_a}$ indicate the X,Y,Z coordinates and attributes mode, respectively. $\mathbf{R}$ is the real manifold. Points in this voxel are regarded as entries in the tensor, which are arranged as $\mathrm{r}_{i_x i_y i_z i_a}$, where $i_x = 1, \ldots, \mathrm{I_x}; \mathrm{i_y} = 1, \ldots, \mathrm{I_y}; \mathrm{i_z} = 1, \ldots, \mathrm{I_z}; i_a = 1, \ldots, \mathrm{I_a}$. The voxel size is defined as 1m and tensor size is defined as $10 \times 10 \times 10 \times 10$. It means that the voxel is equally partitioned as 10 intervals along X,Y,Z coordinate, and 10 attributes contained in each point. Therefore, the entries are the attributes of each point, which are accessed via $\mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z}, \mathrm{I_a}$ indices. That means, attributes are spatially constrained along local direction and implicitly exploited by tensor representation.

The tensor can be represented in terms of its factors using the Tucker model, which is shown as equation(2).

$$\boldsymbol{T} \approx \boldsymbol{X} \times_1 U^{(\mathrm{x})} \times_2 U^{(\mathrm{y})} \times_3 U^{(\mathrm{z})} \times_4 U^{(\mathrm{a})} \qquad (2)$$

Here, $\mathrm{U}^{(\mathrm{x})} \in \mathbf{R}^{\mathrm{I_x} \times \mathrm{J_x}}$, $\mathrm{U}^{(\mathrm{y})} \in \mathbf{R}^{\mathrm{I_y} \times \mathrm{J_y}}$, $U^{(\mathrm{z})} \in \mathbf{R}^{\mathrm{I_z} \times \mathrm{J_z}}$, $\mathrm{U}^{(\mathrm{a})} \in \mathbf{R}^{\mathrm{I_a} \times \mathrm{J_a}}$ are the factor matrices and contain the basis vectors on X coordinate, Y coordinate, Z coordinate and attribute mode. $\boldsymbol{X} \in \mathbf{R}^{\mathrm{J_x} \times \mathrm{J_y} \times \mathrm{J_z} \times \mathrm{J_a}}$ is the core tensor, where $\mathrm{J_x}, \mathrm{J_y}, \mathrm{J_z}, \mathrm{J_a} \leq \mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z}, \mathrm{I_a}$, and its entries show the level of interaction between the different components (Tamara et al., 2007). As such $\mathrm{J_x}, \mathrm{J_y}, \mathrm{J_z}, \mathrm{J_a} \leq \mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z}, \mathrm{I_a}$, the original tensor can be well recovered with the core tensor and a few basis vectors on each mode.

Tucker mode can be written as Kronecker representation: the two representations are equivalent. Let $\otimes$ denotes the Kronecker product, t is the vectorized version of tensor $\boldsymbol{T}$, x is the vectorized version of tensor $\boldsymbol{X}$. The equal Kronecker representation is shown as following:

$$t = \left(U^{(\mathrm{x})} \otimes U^{(\mathrm{y})} \otimes U^{(\mathrm{z})} \otimes U^{(\mathrm{a})}\right) x \qquad (3)$$

2.3 Dictionary Learning

For a set of LiDAR tensors $\{\boldsymbol{T}_k\}_{k=1}^{\mathrm{K}}$, where $\boldsymbol{T}_k \in \mathbf{R}^{\mathrm{I_x} \times \mathrm{I_y} \times \mathrm{I_z} \times \mathrm{I_a}}$ is 4-order point tensor and K is number of tensors. Dictionaries $D = [D^{\mathrm{x}}, D^{\mathrm{y}}, D^{\mathrm{z}}, D^{\mathrm{a}}]$, and $D^{\mathrm{x}}, D^{\mathrm{y}}, D^{\mathrm{z}}, D^{\mathrm{a}}$ are dictionaries on X coordinate, Y

coordinate, Z coordinate and attribute mode, respectively. Tensor dictionary learning aims to calculate the dictionaries $D = [D^{\mathrm{x}}, D^{\mathrm{y}}, D^{\mathrm{z}}, D^{\mathrm{a}}]$ and sparse tensor $\{\boldsymbol{X}_k\}_{k=1}^{\mathrm{K}}$ by following model. ($\|\cdot\|_2$denotes the $l_2 - norm$).

$$\min_{D^{\mathrm{x}}, D^{\mathrm{y}}, D^{\mathrm{z}}, D^{\mathrm{a}}} \sum_{k=1}^{\mathrm{K}} \|\boldsymbol{T}_k - \boldsymbol{X}_k \times_1 D^{\mathrm{x}} \times_2 D^{\mathrm{y}} \times_3 D^{\mathrm{z}} \times_4 D^{\mathrm{a}}\|_2 \tag{4}$$

The dictionary learning can be performed independently for each class to build a sub-dictionary. Denoted by $D_{\mathrm{i}}^{\mathrm{x}}, D_{\mathrm{i}}^{\mathrm{y}}, D_{\mathrm{i}}^{\mathrm{z}}, D_{\mathrm{i}}^{\mathrm{a}}$ are sub-dictionaries associated with class i on X coordinate, Y coordinate, Z coordinate and attribute mode, class $\mathrm{i} = [1,2,\dots,\mathrm{c}]$. Let $\{\boldsymbol{T}_{\mathbf{ik}}\}_{\mathrm{k}=1}^{\mathrm{K}}$ be the training tensor set from class i, and K is number of tensors belong to class i. The sub-dictionaries from each class $D_{\mathrm{i}}^{\mathrm{x}}, D_{\mathrm{i}}^{\mathrm{y}}, D_{\mathrm{i}}^{\mathrm{z}}, D_{\mathrm{i}}^{\mathrm{a}}$ can be learned from model:

$$\min_{D_{\mathrm{i}}^{\mathrm{x}}, D_{\mathrm{i}}^{\mathrm{y}}, D_{\mathrm{i}}^{\mathrm{z}}, D_{\mathrm{i}}^{\mathrm{a}}} \sum_{\mathbf{k}=1}^{\mathbf{K}} \left\|\boldsymbol{T}_{\mathrm{ik}} - \boldsymbol{X}_{\mathrm{ik}} \times_1 D_{\mathrm{i}}^{\mathrm{x}} \times_2 D_{\mathrm{i}}^{\mathrm{y}} \times_3 D_{\mathrm{i}}^{\mathrm{z}} \times_4 D_{\mathrm{i}}^{\mathrm{a}}\right\|_{\mathbf{2}} \tag{5}$$

Equation (5) can be solved by the Tucker decomposition based on equation (2).

Every training point tensor $\boldsymbol{T}_{\mathbf{ik}}$ from class *i* is tucker decomposed to get the $U^{(\mathrm{x})}, U^{(\mathrm{y})}, U^{(\mathrm{z})}, U^{(\mathrm{a})}$, then a certain number of basis vectors of $U^{(\mathrm{x})}, U^{(\mathrm{y})}, U^{(\mathrm{z})}, U^{(\mathrm{a})}$ are added into dictionaries $D_{\mathrm{i}}^{\mathrm{x}}, D_{\mathrm{i}}^{\mathrm{y}}, D_{\mathrm{i}}^{\mathrm{z}}, D_{\mathrm{i}}^{\mathrm{a}}$ .The final dictionaries $D^{\mathrm{x}}, D^{\mathrm{y}}, D^{\mathrm{z}}, D^{\mathrm{a}}$ are described as following, c is the number of classes.

$$D^{\mathrm{x}} = [D_1^{\mathrm{x}}, D_2^{\mathrm{x}}, \dots D_{\mathrm{c}}^{\mathrm{x}}];$$
$$D^{\mathrm{y}} = [D_1^{\mathrm{y}}, D_2^{\mathrm{y}}, \dots D_{\mathrm{c}}^{\mathrm{y}}];$$
$$D^{\mathrm{z}} = [D_1^{\mathrm{z}}, D_2^{\mathrm{z}}, \dots D_{\mathrm{c}}^{\mathrm{z}}];$$
$$D^{\mathrm{a}} = [D_1^{\mathrm{a}}, D_2^{\mathrm{a}}, \dots D_{\mathrm{c}}^{\mathrm{a}}];$$

Algorithm: Tensor OMP

Require: input point tensor $\boldsymbol{T} \in \mathrm{R}^{I_1 \times I_2 \times I_3 \times I_4}$, Dictionaries$D^{\mathrm{x}} \in R^{I_1 \times J_1}$, $D^{\mathrm{y}} \in R^{I_2 \times J_2}$, $D^{\mathrm{z}} \in R^{I_3 \times J_3}$, $D^{\mathrm{a}} \in R^{I_4 \times J_4}$, maximum number of non-zeros coefficients *k* in each mode.

Output: sparse tensor $\boldsymbol{X}$, non-zeros coefficients index in sparse tensor (M_1, M_2, M_3, M_4)

Step:

1, initial: $M_n = [\emptyset](n = 1,2,3,4)$, Residual $\boldsymbol{R} = \boldsymbol{T}$, $\boldsymbol{X} = 0$, *k*=0, $t = vec(\boldsymbol{T})$

2,while $\|M_n\|_0 \leq k$ do

3,$[m_1, m_2, m_3, m_4] = \arg max_{[m_1, m_2, m_3, m_4]} \left|\boldsymbol{R} \times_1 D^{\mathrm{x}T}(:, m_1) \times_2 D^{\mathrm{y}T}(:, m_2) \times_3 D^{\mathrm{z}T}(:, m_3) \times_4 D^{\mathrm{a}T}(:, m_4)\right|$

4, $M_n = M_n \cup [m_1, m_2, m_3, m_4]$ $(n = 1,2,3,4)$. $TD^{\mathrm{x}} = D^{\mathrm{x}}$ $(:, M_1)$, $TD^{\mathrm{y}} = D^{\mathrm{y}}$ $(:, M_2)$, $TD^{\mathrm{z}} = D^{\mathrm{z}}$ $(:, M_3)$, $TD^{\mathrm{a}} = D^{\mathrm{a}}$ $(:, M_4)$;

5, $x = \arg\ min_u \|(TD^{\mathrm{a}} \otimes TD^{\mathrm{z}} \otimes TD^{\mathrm{y}} \otimes TD^{\mathrm{x}})u - t\|_2^2$;

6, $\boldsymbol{X} = \boldsymbol{tensorize}(x)$;

6, $\boldsymbol{R} = \boldsymbol{T} - \boldsymbol{X} \times_1 \mathrm{TD1} \times_2 \mathrm{TD2} \times_3 \mathrm{TD3} \times_4 \mathrm{TD4}$;

7, t=t+1;

8, end while

9,return $\boldsymbol{X}$, (M_1, M_2, M_3, M_4)

2.4 Tensor Sparse Representation for Classification

The objective of tensor sparse coding is to find a sparse representation of a tensor $\boldsymbol{T}$ with respect to the factors $\boldsymbol{D}$ on each mode. This means that the sparse coding is obtained by solving following optimization model:

$$\min_{\boldsymbol{X}} \sum_{\mathbf{k}=1}^{\mathbf{K}} \|\boldsymbol{T} - \boldsymbol{X} \times_1 D^{\mathrm{x}} \times_2 D^{\mathrm{y}} \times_3 D^{\mathrm{x}} \times_4 D^{\mathrm{x}}\|_{\mathbf{2}} \quad \boldsymbol{subject\ to}\ \|\boldsymbol{X}\|_{\mathbf{0}} \leq \boldsymbol{n} \tag{6}$$

$\|\boldsymbol{X}\|_2$ denotes the $l_0 - norm$ of tensor $\boldsymbol{X}$, which is also considered as the sparsity of tensor $\boldsymbol{X}$. The problem is presented as minimizing the approximation error within a certain sparsity level, which can be approximately solved by greedy pursuit algorithms

such as Orthogonal Matching Pursuit (OMP). Classical OMP locates the support of the sparse vector that have best approximation of sample data from dictionary. It selects the support set by one index at each iteration until K atoms are selected or the approximation error is within a preset threshold (Chen et al. 2011), where K is the sparsity. We use the steps of the classical OMP algorithm for tensors as it is shown in Algorithm TensorOMP. The algorithm is proposed by Caiafa and Cichocki (Caiafa and Cichocki, 2012).

In the step 5 of the algorithm, $TD^{\mathrm{x}}, TD^{y}, TD^{\mathrm{z}}, TD^{\mathrm{a}}$ correspond to the sub-dictionaries obtained by restricting the n-mode dictionaries to the columns indicated by indices M_n.

3. EXPERIMENT

3.1 Data description

We perform the classification on two sections of airborne LiDAR dataset of Vienna city. The density of datasets mostly range from 8 to 75 points /m^2. The area of both dataset is 100×100 m. The datasest1 is with flat terrain and contains 710870 points. The dataset2 has more complex environment and 817939 points in total. Both datasets contains complex objects like buildings with various height and shape, single trees, grouped and low vegetation, hedges, fences, cars and telegraph poles. In the classification procedure, the objects are categorized into 5 classes: *open ground* which is uncovered or not blocked by any objects; *building roof*; *vegetation*; *covered ground* which is usually under the high trees or building roof; *building wall*. However, in the evaluation session the open ground and covered ground are merged into ground to achieve the overall ground detection accuracy.

To build the tensors, the neighborhood threshold is defined as 1 meter for selecting points into the voxel, then the voxel is represented as a 4-order tensor $\boldsymbol{T} \in \mathbf{R}^{10\times10\times10\times10}$. It means that the spatial coordinates of the voxel are regularized into a $10\times10\times10$ cube, and 10 attributes are attached on each point. The entries are the normalized attribute values and accessed via four indices. Fig1 shows the points in the voxel and tensor representation by X, Y, Z coordinate indices. And the 10 attributes are described as following:

(1) Relative height. It is a binary value, which is defined as 1 if the point height above the threshold, otherwise is defined as 0. This is useful for indicating ground and non-ground points.

(2) NormalZ. NormalZ are the normal vectors of local planes in Z direction, which are estimated by points in a small neighborhood.

(3)-(5) Eigenvalue1; Eigenvalue2; Eigenvalue3. The covariance matrix for the normal vectors is computed to find the eigenvalues, which include Eigenvalue1 λ_1; Eigenvalue2 λ_2; Eigenvalue3 $\lambda_3(\lambda_1 > \lambda_2 > \lambda_3)$. λ_2 λ_3 have low values for planar object and higher values for voluminous point clouds. Three structure features derived from eigenvalues are anisotropy, sphericity and planarity, which describe the spatial local points' distribution and defined as following equation (Chehata et al., 2009).

(6) Anisotropy. Anisotropy= $(\lambda_1 - \lambda_3)/\lambda_1$.

(7) Sphericity. Sphericity= $\lambda_3/\lambda_1(\lambda_1\text{-}\lambda_3)/\lambda_1$.

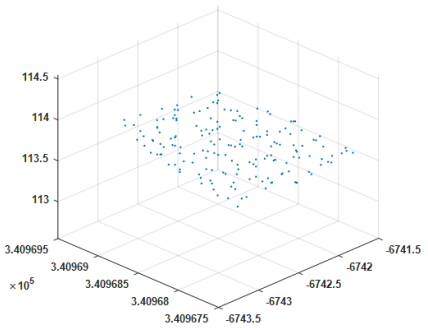
(8) Planarity. Planarity= $(\lambda_2 - \lambda_3)/\lambda_1$.

(9) NormalSigma0. The standard deviation of normal estimation. The value would be high in rough area and low in smooth area.

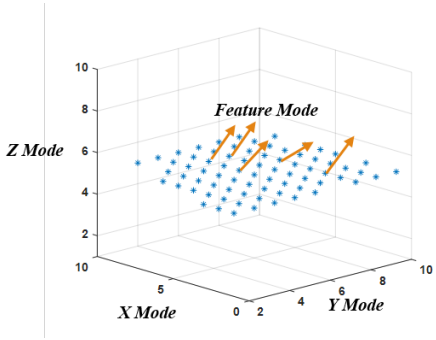
(10) Echo Ratio. the echo ratio is a measure for local transparency and roughness. It is defined as follows(Höfle et al. 2009)

$$\mathrm{ER} = n_{3D}/n_{2D}\times100$$

With $n_{3D} \le n_{2D}$, n_{3D} is the number of neighbors found in a certain search distance measured in 3D and $\mathrm{n}_{2\mathrm{D}}$ is the number of neighbors found in same distance measured in 2D. The ER is nearly 100% for flat surface, whereas the ER decreases for penetrable surface parts since there are more points in a vertical search cylinder than there are points in a sphere with the same radius.



(a) LDAR ground points visualization in the voxel



(b) LiDAR ground points visualized by 4-order tensor form

Figure 1. Point distribution in voxel and tensor

3.2 Classification

The experiments show that the LiDAR tensor can be fully recovered with the compressed core tensor $6\times6\times6\times6$, which means it needs at least 6 basis vectors in each mode and corresponding core tensor for reconstruction. Hence, only 6 basis vectors are added into each sub-dictionary, and the final dictionary would be a 10×30 matrix in each mode.

First of all, the sub-dictionary associated with a specific class i is created. Denote by $D_i^{\mathrm{x}}, D_i^{\mathrm{y}}, D_i^{\mathrm{z}}, D_i^{\mathrm{a}}$ the sub-dictionary associated with class i on mode X,Y,Z and attribute, $class\ i = [1,2,3,4,5]$. 6 training tensors are randomly selected from each class, $\boldsymbol{T}_{\mathrm{i}}^{\mathrm{k}}$ donates the k-th training tensor in class i, $k = 1,2\ldots6$. Training tensor $\boldsymbol{T}_{\mathrm{i}}^{\mathrm{k}}$ is decomposed by Tucker model to get basis vectors in each mode:

$$\boldsymbol{T}_{\mathrm{i}}^{\mathrm{k}} \approx \boldsymbol{X}_{\mathbf{i}}^{\mathrm{k}} \times_1 U_{\mathrm{i}}^{(\mathrm{x})} \times_2 U_{\mathrm{i}}^{(\mathrm{y})} \times_3 U_{\mathrm{i}}^{(\mathrm{z})} \times_4 U_{\mathrm{i}}^{(\mathrm{a})} \quad (7)$$

The first column in matrix $U_{\mathrm{i}}^{(\mathrm{x})}, U_{\mathrm{i}}^{(\mathrm{y})}, U_{\mathrm{i}}^{(\mathrm{z})}, U_{\mathrm{i}}^{(\mathrm{a})}$ is added into the corresponding sub-dictionary in each mode. Thus, $D_i^{\mathrm{x}}, D_i^{\mathrm{y}}, D_i^{\mathrm{z}}, D_i^{\mathrm{a}}$ can be described as :

$$D_i^{\mathrm{x}} = \left[U_1^{(\mathrm{x})}(:,1), U_2^{(\mathrm{x})}(:,1) \ldots U_n^{(\mathrm{x})}(:,1)\right]$$

$$D_i^{\mathrm{y}} = \left[U_1^{(\mathrm{y})}(:,1), U_2^{(\mathrm{y})}(:,1) \ldots U_n^{(\mathrm{y})}(:,1)\right]$$

$$D_i^{\mathrm{z}} = \left[U_1^{(\mathrm{z})}(:,1), U_2^{(\mathrm{z})}(:,1) \ldots U_n^{(\mathrm{z})}(:,1)\right]$$

$$D_i^{\mathrm{a}} = \left[U_1^{(\mathrm{a})}(:,1), U_2^{(\mathrm{a})}(:,1) \ldots U_n^{(\mathrm{a})}(:,1)\right]$$

And the final dictionary on each mode can be represented as following:

$$D^{\mathrm{x}} = [D_1^{\mathrm{x}}, D_2^{\mathrm{x}}, D_3^{\mathrm{x}}, D_4^{\mathrm{x}}, D_5^{\mathrm{x}}]$$
$$D^{y} = [D_1^{\mathrm{y}}, D_2^{\mathrm{y}}, D_3^{\mathrm{y}}, D_4^{\mathrm{y}}, D_5^{\mathrm{y}}]$$
$$D^{\mathrm{z}} = [D_1^{\mathrm{z}}, D_2^{\mathrm{z}}, D_3^{\mathrm{z}}, D_4^{\mathrm{z}}, D_5^{\mathrm{z}}]$$
$$D^{\mathrm{a}} = [D_1^{\mathrm{a}}, D_2^{\mathrm{a}}, D_3^{\mathrm{a}}, D_4^{\mathrm{a}}, D_5^{\mathrm{a}}]$$

3.3 Classification Result And Discussion

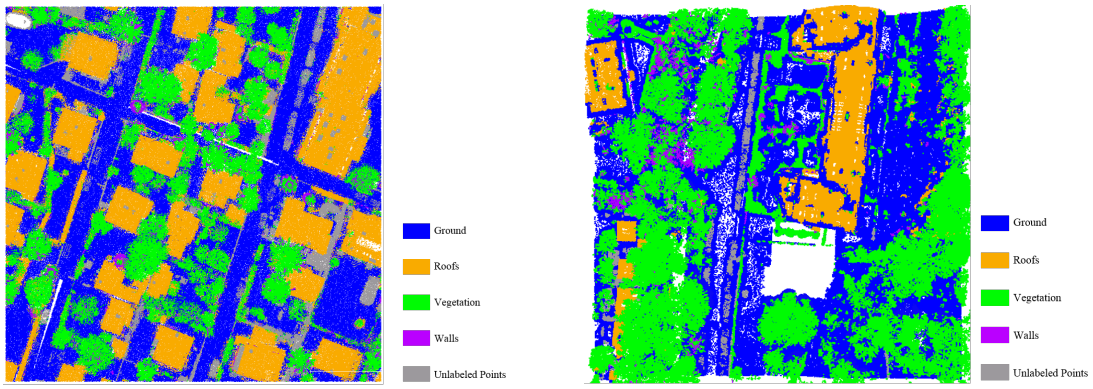
Visual inspection indicates that most objects are detected correctly in Fig2. The overall classification accuracy is 82% for dataset1 and 80% for dataset2. Tab1 and Tab2 are the confusion matrices which demonstrates prediction ability of the algorithm on various objects.

Some buildings and trees are extracted from dataset1 and dataset2 for error points analysis. Fig3(a) and (e) indicate that some parts of boundary points in roof in dataset2 are misclassified into ground(12.3%), but the algorithm performs very well in identifying roofs with dataset1, which achieve a high accuracy of 98.3%. 8% of vegetation are wrongly predicted as walls in dataset1, which is mainly caused by trees with a vertical structure or high pruned and trimmed trees (Fig 3(c) and (g)). And 6.9% of vegetation are misclassified into roofs in dataset1. This usually occurred on low flat vegetation which has similar attributes with roofs, examples can be found in Fig3 (d) and (f). Wall points are usually located in a more complex scenario. In Fig4 (a) and (e), it is a balcony wall and ends up confused with vegetation. In Fig4 (b) and (f), top and bottom wall points are labeled as roof and ground due to the close location to roof and ground, but middle part of walls can obtain correct labels. Thus, this algorithm works well in distinguishing objects with clear spatial structures.

Considering that only 30 training points are used, this tensor SRC algorithm can achieve an overall good performance, especially in roof identification.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-IV-2-W4-107-2017 | © Authors 2017. CC BY 4.0 License.

However, high accuracy cannot be obtained in complex scene, such as wall boundary identification from roof and ground. Since the main part objects could be correctly labeled, the error points can be reduced by further filtering method.



(a) Dataset1 classification result (b) Dataset2 classification result

Figure 2. Classification result

Table 1. Dataset1 Classification confusion matrix

Class	Ground	Roofs	Veg	Walls
Ground	89.0%	7.0%	2.2%	1.8%
Roofs	0.3%	98.3%	1.1%	0.2
Veg	5.1%	6.9%	79.7%	8.3%
Walls	17.9%	15.5%	9.7%	57.0%

Table 2. Dataset2 Classification confusion matrix

Class	Ground	Roofs	Veg	Walls
Ground	79.5%	3.0%	15.3%	2.15%
Roofs	12.3%	84.3%	3.4%	0%
Veg	7.5%	2.2%	85.8%	4.6%
Walls	4.9%	3.8%	20.9%	70.4%

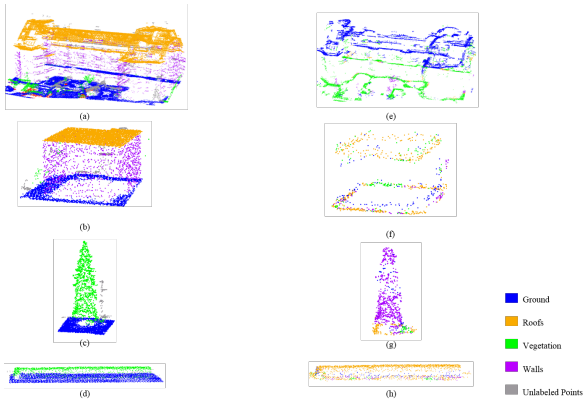


Figure 3. Reference and misclassified point in 4 scenes

4. CONCLUSION

A tensor sparse representation classification method has been proposed and tested with real airborne LiDAR data. The method integrates spatial distribution and attributes by tensor representation. Only 6 training points from each class are utilized to build the dictionary. It achieves an overall classification accuracy of 82%. This algorithm has respectable performance in distinguishing object with clear shape pattern. Further work will focus on the dictionary improvement based on dictionary learning algorithm, which can distinguish more minor and unambiguous objects.

5. ACKNOWLEDGEMENT

The LiDAR data comes from Vienna city government. The project is supported by National Natural Science Foundation of China (Project No.41371333).

6. REFERENCE

Anguelov, D., B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz and A. Ng., 2005. "Discriminative learning of markov random fields for segmentation of 3d scan data". Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, IEEE.

Caiafa, C. F. and A. Cichocki., 2012. "Block sparse representations of tensors using Kronecker bases". Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, IEEE.

Chehata, N., L. Guo and C. Mallet., 2009. Airborne lidar feature selection for urban classification using random forests. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **38**(Part 3): W8.

Chen, Y., N. M. Nasrabadi and T. D. Tran., 2011. Hyperspectral image classification using dictionary-based sparse representation. *IEEE Transactions on Geoscience and Remote Sensing* **49**(10): 3973-3985.

Höfle, B., W. Mücke, M. Dutter, M. Rutzinger and P. Dorninger., 2009. "Detection of building regions using airborne lidar—A new combination of raster and point cloud based GIS methods". Proceedings of GI-Forum 2009—International Conference on Applied Geoinformatics.

Huang, K. and S. Aviyente., 2006. "Sparse representation for signal classification". NIPS.

Im, J., J. R. Jensen and M. E. Hodgson., 2008. Object-based land cover classification using high-posting-density LiDAR data. *GIScience & Remote Sensing* **45**(2): 209-228.

Joachim Niemeyer, F. R., Uwe Soergel., 2013. "Classification of Urban LiDAR data using Conditional Random Field and Random Forests". Proc. 7th IEEE/GRSS/ISPRS Joint Urban Remote Sens.

Kolda, T. G. and B. W. Bader., 2009. Tensor decompositions and applications. *SIAM review* **51**(3): 455-500.

Lee, Y. K., C. Y. Low and A. B. J. Teoh., 2015. "Tensor kernel supervised dictionary learning for face recognition". Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific, IEEE.

Li, N., C. Liu, N. Pfeifer, J. F. Yin, Z. Y. Liao and Y. Zhou. 2016. TENSOR MODELING BASED FOR AIRBORNE LiDAR DATA CLASSIFICATION. Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci. XLI-B3: 283-287.

Najafi, M., S. T. Namin, M. Salzmann and L. Petersson., 2014. "Non-associative higher-order markov networks for point cloud classification". European Conference on Computer Vision, Springer.

Niemeyer, J., F. Rottensteiner and U. Soergel., 2014. Contextual classification of lidar data and building object detection in urban areas. *ISPRS journal of photogrammetry and remote sensing* **87**: 152-165.

Peng, Y., D. Meng, Z. Xu, C. Gao, Y. Yang and B. Zhang., 2014. "Decomposable nonlocal tensor dictionary learning for multispectral image denoising". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Renard, N. and S. Bourennane., 2009. Dimensionality reduction based on tensor modeling for classification methods. *Geoscience and Remote Sensing, IEEE Transactions* on **47**(4): 1123-1131.

Roemer, F., G. Del Galdo and M. Haardt., 2014. "Tensor-based algorithms for learning multidimensional separable dictionaries". Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, IEEE.

Wright, J., A. Y. Yang, A. Ganesh, S. S. Sastry and Y. Ma., 2009. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis* and machine intelligence 31(2): 210-227.

Yang, S., M. Wang, P. Li, L. Jin, B. Wu and L. Jiao. 2015. Compressive hyperspectral imaging via sparse tensor and nonlinear compressed sensing. IEEE Transactions on Geoscience and Remote Sensing 53(11): 5943-5957.