Automatic Generation of as-built Geometric Civil Infrastructure Models from Point Cloud Data

G. Zhang[1], P. A. Vela[1] and I. Brilakis[2]

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0250; email: zhanggc@gatech.edu, pvela@gatech.edu
[2]Department of Engineering, University of Cambridge BC2-07, Trumpington Street, Cambridge, CB2 1PZ, UK; email: ib340@eng.cam.ac.uk

ABSTRACT

Converting remote sensing point cloud data (PCD) into solid CAD models consisting of civil infrastructure components is a crucial step in generating the as-built building information models. Previous research has enabled automatic generation of surface primitives from raw PCDs. However, the fully automatic conversion from surface primitives to infrastructure component models remains an unsolved problem. In this work, an automatic and linear-runtime approach is presented which generates the as-built infrastructure component models by recognizing the solid CAD entities and learning the infrastructure component labels from the fitted surface primitives. The algorithm utilizes a decision tree with the following decision variables: the type, parametric model, orientation, and mutual geometric relations of the fitted surface primitives. The decision tree is trained with easily generated synthetic data and is applied to query real-world data with complexity O(1). The output of the solid entities includes cuboid, cylinder, ball, etc. and the infrastructure component labels such as columns, caps, deck, beams, etc. The algorithm is tested with various PCDs modeling real bridges.

INTRODUCTION

Changes during the construction phase and the operating phase of the infrastructure life cycle can be represented in as-built building information models (BIMs) to enable the representation of actual states of an existing facility more truthfully than the as-designed documents. However, the conventional process of generating an as-built BIM is a time-consuming manual process. Therefore, recent research has been focused on the automatic generation of as-built BIMs. Typically, the generation process of an as-built BIM takes the remote sensing point cloud data (PCD) as input. Then geometric models are extracted from the raw PCDs. Besides the geometric models, different components of the facility are recognized and the components' descriptions are generated. Finally, the geometric models combined with the facility component information are output into industrial data formats, e.g. Industry Foundation Classes (IFC) data models.

To automate the generation of as-built BIMs, both extracting geometric models from raw PCDs and recognizing the facility components must be automated. The problem of automatically extracting the geometric primitives has been largely solved, either by extracting the planar patches from PCDs (Zhang, G., et al, 2012), or by extracting surface/shape primitives from the point clouds (Zhang, G., et al 2013; Schnabel, R., et al 2007; Li, Y., et al 2011). Although there is a solution to semantic learning for indoor environments (Xiong, X., et al 2013), no solution exists for recognizing components and generating semantic models of infrastructures like bridges. We address this problem by proposing a novel method to recognize both the semantic labels of facility components (e.g. beams, deck, columns, etc.) and geometric entity labels of computer-aided design (CAD) models (e.g. cuboids, cylinders, sheet, etc.). Note that there is not necessarily a one-to-one mapping between these two kinds of labels. We tested the method on bridge PCDs, however the same principles and method should work for other structures as well (e.g. parking decks, building skeletons, etc.). The method takes the extracted surface primitives as input, then classifies the primitives into different classes of components or entities, and finally generates the CAD and IFC models with both the geometric information and semantic labels. The algorithm is designed for fast application: the classification step is of linear runtime. The algorithm is tested with PCDs modeling real-world bridges, and evaluated based on the classification error compared to the ground truth. Both evaluated results reveal the effectiveness of the algorithm.

RELATED WORK

The work in this paper is related to 3D entity models extraction from point cloud data. In 1993, (Arman, F., et al 1993) proposed a method to recognize CAD models from range images by generating a strategy to select models' geometric features in sequence for identifying and localizing the model in the scene. The strategy is guided by objects' visibility, detectability, frequency of occurrence, etc, and the file output is stored in standard CAD models. Different from this, recent research in the computing in civil engineering community mainly studies the problem of entity model recognition for as-built BIMs. In (Bosché, F., et al, 2008, 2010), a method based on an Iterative Closest Point (ICP)-based fine registration is proposed for recognizing CAD model objects from laser scans. Similarly, (Kim, C, et al 2011) also addresses the matching between the point clouds and the CAD models. They target the application of more complicated CAD models with registration combining principal component analysis coarse registration, and ICP fine registration. Another series of work aimed at as-built BIMs is presented in (Xiong, X., et al 2013), which utilizes supervised stacked learning to learn indoor environments. The data-driven approach relies on high-quality and large training datasets, which themselves are difficult to obtain. Moreover, it is for indoor surroundings, not for the outdoor infrastructure that we target.

Different from the existing methods, our approach builds upon the previous work for shape primitive extraction (Zhang, G., et al 2013). The retrieval of the model entities is done by classification with the primitive information instead of

performing registration, which makes our method able to execute with linear runtime, much more efficiently than the existing methods.

PROPOSED METHOD

The proposed method consists of four major algorithms:
(1) Pre-processing step: to extract the surface primitives and the support given an input query PCD, in order to generate classification features in later steps.
(2) Feature extraction step: to generate the proposed feature vectors, which capture the distinct geometric properties of each component to facilitate classification.
(3) Classification step: to classify each surface primitive and generate the labels for both the facility component labels and geometric entity labels, with a pre-trained multiple-class adaboost decision tree.
(4) Model generation and output step: based on the labels generated from classifier, this step generates the final CAD model files and IFC files for as-built BIMs.

The following flowchart illustrates the structure of the proposed algorithm. Although the proposed algorithm is generic and can be applied to different facilities, we focus on the application to bridge point cloud data.

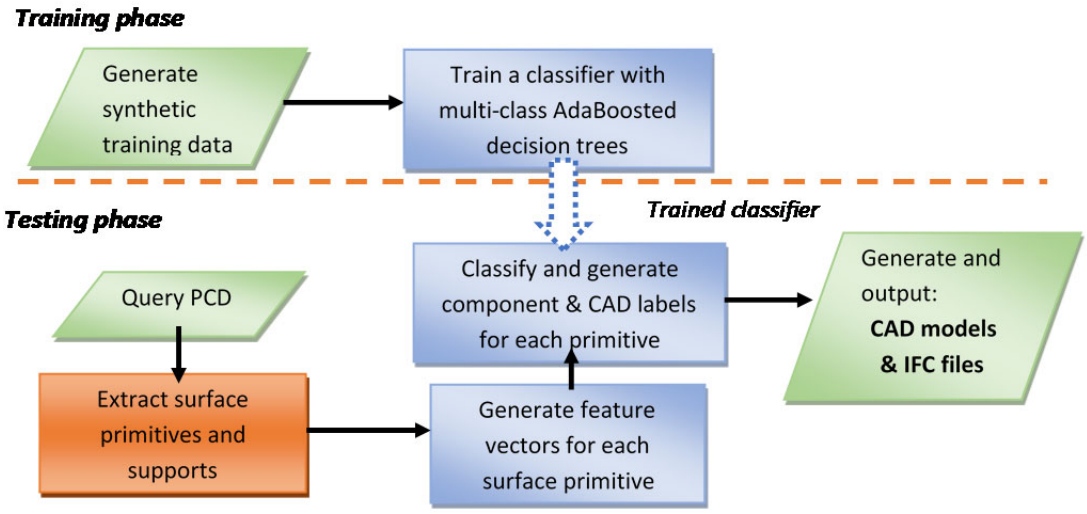

Figure 1. Pipeline of the proposed method; blue blocks represent steps in the new algorithms; red blocks represent steps using existing algorithms.

Feature vectors of PCDs for classification.

Features for classification. Given the input data consisting of multiple surface primitives $\{\mathcal{M}_i\}_{i=1}^{M}$ and the corresponding support set $\{\mathcal{S}_i\}_{i=1}^{M}$ for the primitives, a set of feature vector $\{f_i\}_{i=1}^{M}$ is generated with one feature vector corresponding to one extracted primitive. The feature vector f_i is a $(9 + 2N_p)$ dimensional vector consisting of the coordinates:

$$f_i = [T_i, n_i, V_i, B_i, L_i],$$

where N_p is the number of the primitive types. This feature vector captures the following information of each input primitive: primitive type, principal direction,

normal vector, spatial scale and neighborhood statistics. The details of each sub-vector are elaborated as follows.

(1) $T_i \in \mathbb{N}^{N_p}$ is an index encoding the type of the surface primitive. For example, in the case of a bridge whose surface primitive set has planes and cylinders, then T_i can be encoded with indices $\{1, 2\}$ as:

$$T_i = \begin{cases} 1, & if\ surface\ is\ a\ plane \\ 2, & if\ surface\ is\ a\ cylinder \end{cases}$$

(2) $n_i \in \mathbb{R}^3$ is the normal of each surface primitive. This corresponds to the coefficients of the x-term, y-term and z-term in the estimated surface primitive model.

(3) $V_i \in \mathbb{R}^3$ is a unit vector which captures the principal direction of the support of the primitive $\mathcal{M}_i$. Given the support of primitive $\mathcal{M}_i$ as $\mathcal{S}_i \in \mathbb{R}^{3\times k}$, $\mathcal{S}_i$ is first centered by it mean, yielding $\overline{\mathcal{S}_i}$, then V_i is computed by finding the eigenvector corresponding to the largest eigenvalue from $\overline{\mathcal{S}_i}^T \overline{\mathcal{S}_i}$. Note that since we only need the first principal eigenvector, performing a power iteration is sufficient instead of performing a full spectral decomposition, where the latter one is of much more computational cost.

(4) $B_i \in \mathbb{R}^3$ is a vector capturing the relative scale of bounding box size $\mathcal{S}_i$ compared to the bounding box size of the whole point set. Assuming the whole point set is $\{P_i = (x_i, y_i, z_i)\}$, then the bounding box size of $\{P_i\}$ is given by $[a, b, c] = [(\max(x_i) - \min(x_i)),\ (max(y_i) - min(y_i)),\ (max(z_i) - min(z_i))]$. If the bounding box of the primitive support $\mathcal{S}_i$ is $[a_i, b_i, c_i]$, then $B_i = \left[\frac{a^k}{a}, \frac{b^k}{b}, \frac{c^k}{c}\right]$.

(5) $L_i \in \mathbb{R}^{N_{p'}}$ is a vector capturing the neighborhood statistics indicating the number of each primitive types which are in 1-distance neighbor (directly connected) of the current surface primitive. For example, in the case of a bridge with only planes and cylinders, a primitive is directly connected to 3 cylinders and 2 other planes, then $L_i = [2, 3]$. The connectivity is determined by thresholding on the single-linkage pairwise Euclidean distance of each support.

Adaboost for classification. Given the synthetic training dataset with ground truth infrastructure labels and CAD entity labels, we first compute the feature vector of each primitive. Then the feature vectors and the training labels are used to train the classifier (in total two classifiers are trained; one for classifying infrastructure component labels and one for CAD entity labels). Considering that the feature vector consists of different variable types (categorical, discrete, and continuous variables) and the classification problem is nonlinear, we proposed to use AdaBoosted decision trees for the classification.

AdaBoost is an iterative procedure which combines many weak classifiers with different weights to approximate the optimal Bayes classifier. In our work, maximum entropy decision trees are used as the weak classifiers. In the training phase, during each AdaBoost iteration, the feature vectors are first used to train one decision tree. Then according to the classification error, the classifier is weighted by a weight learnt by AdaBoost algorithm with an exponential loss function. Specifically, for the multi-class problem in our case, we use the SAMME algorithm (Zhu, J., et al 2009), which utilizes the population minimizer of multi-class exponential loss. Given the input feature set $\{f_i\}$ containing K classes, the algorithm starts by initializing all

the weights for weak classifiers as $w_k = \frac{1}{K}$. Then if N_C weak classifiers, i.e. the decision trees, will be boosted, in total N_C iterations need to be conducted. In each iteration, the algorithm first induces a decision tree $\mathcal{T}^{(n_c)}(f)$ with the training set and the initial weight $1/K$. Then the empirical error rate is computed as

$$Err^{(n_c)} = \frac{\sum_{k=1}^{K} w_k \mathbb{I}\left(l_k \neq \mathcal{T}^{(n_c)}(f_k)\right)}{\sum_{k=1}^{K} w_k}$$

where $\mathbb{I}(\cdot)$ is an indicator function: if $l_k \neq \mathcal{T}^{(n_c)}(f_k)$ is true, which means the classified label of $\mathcal{T}^{(n_c)}$ on f_i does not agree with the true label l_k, then $\mathbb{I}\left(l_k \neq \mathcal{T}^{(n_c)}(f_k)\right) = 1$; otherwiser $\mathbb{I}\left(l_k \neq \mathcal{T}^{(n_c)}(f_k)\right) = 0$. With the error rates, the weights are updated as

$$\alpha^{(n_c)} = \left(\log\frac{1 - Err^{(n_c)}}{Err^{(n_c)}} + \log(N_l - 1)\right)$$

$$w_k \leftarrow w_k \cdot \exp\left[\alpha^{(n_c)} \cdot \mathbb{I}\left(l_k \neq \mathcal{T}^{(n_c)}(f_k)\right)\right]$$

where N_l is the number of classes. All the weights are then normalized before the next iteration. After N_C iterations, the final hypothesis is given as

$$C(f) = \text{argmax}_{l'} \sum_{n_c=1}^{N_C} \alpha^{(n_c)} \cdot \mathbb{I}\left(\mathcal{T}^{(n_c)}(f) = l'\right)$$

Decision Tree Induction. Decision tree is a nonmetric method for classification and regression (Duda et al, 2012). The decision tree used in our work follows binary-tree structures, with each node imposing one splitting criterion on one feature in the feature vector. The outcome of each node corresponds to a decision with respect to the feature of the node and yields a splitting of a subset in the training data. Hierarchically, the full training set is split by the root node, while each successive decision splits the proper subset of the data. The decision trees used as a weak classifier in our multi-class AdaBoost are linear decision trees, which generate decision boundaries aligned with the axis hyper-planes in the $(9 + 2N_{P'})$ dimensional feature space. The testing runtime of a decision tree is linear, and the Adaboosted classifier is of linear runtime.

The training of the decision tree is done by top-down induction. In each iteration a decision attribute is selected and assigned for the next node. Then according to the attribute, the training subset is split, creating the descendants of the node. The induction terminates if all the training data is perfect classified.

The attribute selection criterion used in our work is to choose the attribute which maximizes the information gain, or equivalently achieves the most reduction in entropy. Given a subset of training set $\mathcal{M}' = \{m'\}$ for current node and its corresponding label set $\mathcal{L}' = \{\ell'_i\}_{i=1}^{N'}$, we first measure the entropy before decision as

$$H(\mathcal{M}') = -\sum_{i=1}^{N'} \text{Prob}(m' = \ell'_i) \log_2 \text{Prob}(m' = \ell'_i)$$

Assume that after the binary decision A, set $\mathcal{M}'$ is partitioned into $\mathcal{M}_1'$ and $\mathcal{M}_2'$, and $|\mathcal{M}'| = |\mathcal{M}_1'| + |\mathcal{M}_2'|$, then the entropy reduction is measured by information gain: $Gain(A) = H(\mathcal{M}') - \sum_{j=1}^{2} \frac{|\mathcal{M}_j'|}{|\mathcal{M}'|} H(\mathcal{M}_j')$.

Notice that when the decision A is to split between feature values $a_1, a_2, (a_1 \leq a_2)$, then any value between a_1, a_2 will lead to the same entropy reduction. In this case we select the threshold to be $(a_1 + a_2)/2$.

Classification and Output Generation. Once the multi-class AdaBoosted decision tree classifier is trained with the synthetic training samples, it can be used to classify the query PCD directly.

Two classifiers are trained which correspond to two types of labels: infrastructure component labels and CAD entity labels. The infrastructure component labels the semantic description of the component. For example, in the case of bridge infrastructure, component labels include “deck”, “beam”, “columns”, etc. The CAD entity labels include geometric solid models, e.g. “cuboid”, “cylinder”, “sheet”, etc.

After classification, the CAD models are generated by finding the bounding box of support set $\{\mathcal{S}_j\}$ belonging to the same CAD entity. The CAD models are output in ply format. Similarly, IFC files are written according to the infrastructure component labels and the support set. The files are output with IFC4 specifications (buildingSMART, 2013).

EXPERIMENTAL RESULTS

In the experiment, a synthetic training set and two query PCDs are generated according to the real configurations and dimensions of pier supported concrete bridges. The training sets are of lower density than query PCD. Measurement noise of query PCD is simulated with Gaussian noise of standard deviation of 1cm. The infrastructure component labels include: ‘Pier column’, ‘Pier cap’, ‘Beam’, ‘Barrier’, ‘Deck’, and ‘Wingwall’. CAD entity labels include: ‘Cylinder’, ‘Cuboid’, and ‘Sheet’. In general, the classifying infrastructure component labels is more difficult than classifying the CAD entity labels.

In the training phase, 10 PCDs with the ground truth labels are used for training. Figure 2 demonstrates the classification error of the infrastructure component labels w.r.t. the number of decision tree used. The training classification error drops to 0 when the number of decision trees reaches 51. For robustness, we use 55 decision trees for the final classifier.

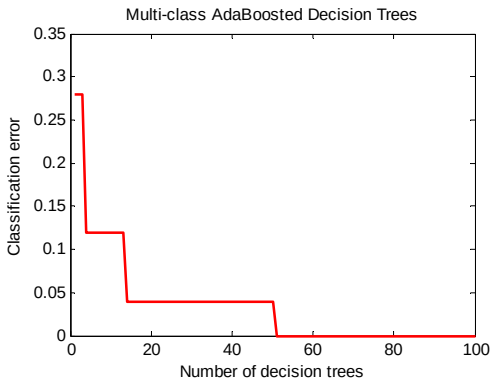

Figure 2. Classification error w.r.t. the number of decision trees used.

With the trained classifier, we classify two query PCDs with the primitives extracted to generate the infrastructure component labels and CAD entity labels. The final output in IFC model combined with semantic labels are shown in Figure 3, in which different components are encoded with different colors.

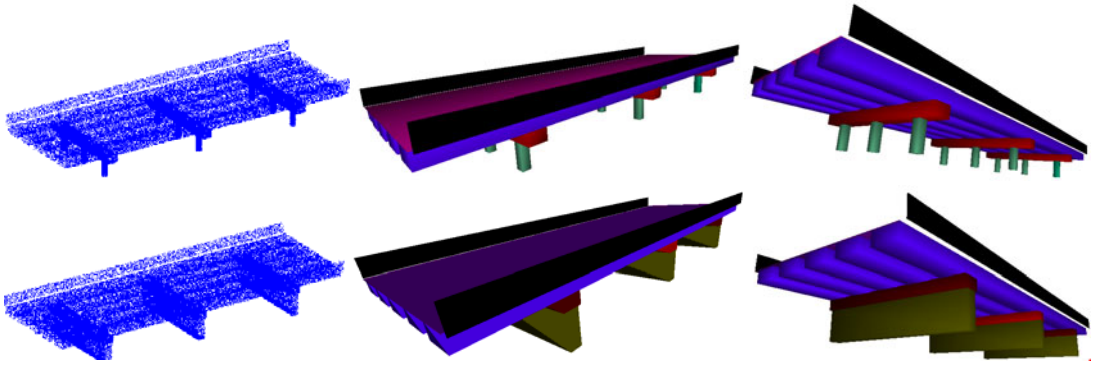

Figure 3. Left top: Query PCDs. Middle and Right: PCD colored according to classified CAD entity labels.

CONCLUSION

In this paper an approach to recognize and classify the infrastructure components for as-built BIMs is proposed. The main contributions are: (1) we propose the feature vectors capturing the distinct geometric properties of the civil components; and (2) based on the features we design a classifier for classifying both infrastructure component labels and geometric entity labels. The algorithm takes a set of found shape primitives and their support as input, generates the corresponding feature vectors, and classifies them with a multi-class AdaBoost Decision Tree, which is trained using a synthetic training set. The final result is output as CAD and IFC files with the infrastructure component labels. Experimental results applying to synthetic bridge data reveal the effectiveness of the algorithm. The future work includes the extension of the approach for other types of civil infrastructures.

REFERENCES

Zhang, G., Karasev, P., Brilakis, I., Vela, P.A. (2012). Sparsity-Inducing Optimization Algorithm for the Extraction of Planar Structures in Noisy Point-Cloud Data. *Proc. IWCCE*, July.

Zhang, G., Vela, P.A., Brilakis, I. (2013). Detecting, Fitting, and Classifying Surface Primitives for Infrastructure Point Cloud Data. *Proc. IWCCE*, July.

Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N. J. (2011). GlobFit: consistently fitting primitives by discovering global relations. In *ACM Transactions on Graphics (TOG)*, 30(4), 52.

Xiong, X., Oliver, A., Akinci, B., Huber, D.(2013), Automatic Creation of Semantically Rich 3D Building Models from Laser Scanner Data, *Automation in Construction*, 31, 325-337.

Arman, F., Aggarwal, J. K. (1993). Cad-based vision: object recognition in cluttered range images using recognition strategies. *CVGIP: Image Understanding*, 58(1), 33-48.

Bosché, F., Haas, C. T. (2008). Automated retrieval of 3D CAD model objects in construction range images. *Automation in Construction*, 17(4), 499-512.

Bosché, F. (2010). Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced engineering informatics*, 24(1), 107-118.

Kim, C., Lee, J., Cho, M., Kim, C. (2011). Fully automated registration of 3D CAD model with point cloud from construction site. In *28th International Symposium on Automation and Robotics in Construction*, 917-922.

Schnabel, R., Wahl, R., Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. In *Computer Graphics Forum* , 26(2), 214-226.

Duda, R. O., Hart, P. E., Stork, D. G. (2012). Pattern classification. *John Wiley & Sons.*

Zhu, J., Zou, H., Rosset, S., Hastie, T. (2009). Multi-class adaboost. *Statistics and Its Interface* , 2, 349–360.

buildingSMART (2013), IFC4 officially released, *URL: http://www.buildingsmart-tech.org/news/ifc4-officially-released.*