Three-Dimensional Object Matching in Mobile Laser Scanning Point Clouds

**Article** in IEEE Geoscience and Remote Sensing Letters - February 2015
DOI: 10.1109/LGRS.2014.2347347

**CITATIONS** 8

**READS** 98

5 **authors**, including:

- **Yongtao Yu**
  Huaiyin Institute of Technology
  41 PUBLICATIONS 370 CITATIONS
  [SEE PROFILE]

- **Jonathan Li**
  University of Waterloo
  252 PUBLICATIONS 3,240 CITATIONS
  [SEE PROFILE]

- **Haiyan Guan**
  Nanjing University of Information Science & …
  52 PUBLICATIONS 428 CITATIONS
  [SEE PROFILE]

- **Cheng Wang**
  Xiamen University
  157 PUBLICATIONS 894 CITATIONS
  [SEE PROFILE]

Some of the authors of this publication are also working on these related projects:

- **Backpacked mobile mapping system for indoor environment** View project
- **Lidar Point Cloud Feature Extraction** View project

All content following this page was uploaded by Haiyan Guan on 07 May 2015.
The user has requested enhancement of the downloaded file.
Three-Dimensional Object Matching in Mobile Laser Scanning Point Clouds

Yongtao Yu, Jonathan Li, Senior Member, IEEE, Haiyan Guan, Fukai Jia, and Cheng Wang, Member, IEEE

Abstract—This letter presents a 3-D object matching framework to support information extraction directly from 3-D point clouds. The problem of 3-D object matching is to match a template, represented by a group of 3-D points, to a point cloud scene containing an instance of that object. A locally affine-invariant geometric constraint is proposed to effectively handle affine transformations, occlusions, incompleteness, and scales in 3-D point clouds. The 3-D object matching framework is integrated into 3-D correspondence computation, 3-D object detection, and point cloud object classification in mobile laser scanning (MLS) point clouds. Experimental results obtained using the 3-D point clouds acquired by a RIEGL VMX-450 system showed that completeness, correctness, and quality of over 0.96, 0.94, and 0.91 are achieved, respectively, with the proposed framework in 3-D object detection. Comparative studies demonstrate that the proposed method outperforms the two existing methods for detecting 3-D objects directly from large-volume MLS point clouds.

Index Terms—Mobile laser scanning (MLS), object classification, point cloud, 3-D object detection, 3-D object matching.

I. INTRODUCTION

THE acquisition and processing of huge volumes of highly dense and irregularly distributed 3-D point clouds from mobile laser scanning (MLS) systems has received increasing attention over the past few years [1]. Automated detection and extraction of objects from such 3-D point clouds has been a very active research topic in the remote sensing community [2]–[6]. A pairwise 3-D shape context was developed in [2], for modeling geometric features of point cloud objects. Such shape context was successfully applied to extract 3-D light poles and trees from MLS point clouds. Hough forest was proposed in [3] to learn mappings from patch features to object centroids. This method obtained promising results in detecting 3-D light poles, palm trees, cars, and traffic signposts from MLS point clouds. A principal component analysis (PCA)-based approach was introduced in [7], for distinguishing trees and buildings. The identified buildings were further processed to extract building footprints based on an integration of random sample consensus (RANSAC) and PCA. In [8], man-made objects were identified from MLS point clouds using a layer-wise approach that sliced the raw point clouds into three layers and classified seed points into different man-made object categories within each layer. Recently, some other studies have been conducted for the detection and extraction of windows [9], railway tracks [10], roads [11], and vertical walls [12] from MLS point clouds.

Rather than handling highly dense 3-D point clouds, some researchers rasterized the 3-D MLS point clouds into 2-D georeferenced feature images. Therefore, existing image processing methods were adopted to achieve the detection of various objects. In [13], road surface point clouds were first converted into georeferenced intensity images through inverse distance weighted interpolation. Then, road markings were extracted using multithresholding and refined through morphological operations. A marked point process of rectangles and disks was developed in [4], for detecting road manholes and sewer well covers from georeferenced intensity images.

In this letter, we develop a framework for matching 3-D objects in 3-D MLS point clouds, directly. A locally affine-invariant geometric constraint is proposed to improve the performance of point-set-based 3-D object matching. The 3-D object matching framework is embedded into the applications of 3-D correspondence computation, 3-D object detection, and point cloud object classification from 3-D point clouds acquired by a RIEGL VMX-450 system.

II. 3-D OBJECT MATCHING FRAMEWORK

A. Problem Formulation

The problem of object matching from point clouds is defined as matching a group of template feature points that represent an object to a group of scene feature points that represent a scene containing an instance of that object. Here, a feature point is a 3-D point sampled from either the template or the scene. First, a group of feature points is selected from the template and scene point clouds, respectively. Denote $n_T$ and $n_S$ ($n_T \leq n_S$) as the number of template and scene feature points, respectively. Let $T \in R^{n_T \times 3}$ and $S \in R^{n_S \times 3}$ be the matrices recording the 3-D coordinates of the template and scene feature points, respectively. The goal of object matching is to determine a matching function $f(\cdot)$ that 1) matches each template feature point $p_i = (x_i, y_i, z_i) \in R^3$, $i = 1, \ldots, n_T$ to a scene feature point $q_j = (x_j, y_j, z_j) \in R^3$, $j = 1, \ldots, n_S$ and 2) minimizes the following overall objective function containing both feature and geometric matching costs:

$$
\min_{j} \sum_{i=1}^{n_T} \left( c(p_i, f(p_i)) + \lambda \cdot g(p_i, N_{p_i}; f(p_i), N_{f(p_i)}) \right)
$$

where $c(p, q)$ denotes the feature matching cost between point $p$ and point $q$; $N_{p_i}$ is the local neighborhood of point $p_i$; $g(\cdot)$
represents a geometric matching cost function that measures the dissimilarity between two sets of ordered points \( \{p_i, N_{p_i}\} \) and \( \{f(p_i), N_f(p_i)\} \); \( \lambda \) controls the relative weight between the feature and geometric matching costs.

B. Feature Matching Cost

The persistent feature histogram (PFH) [14] has proved to be a robust and highly distinctive feature descriptor for discrete point clouds. A PFH is based on the relationships between the points in the \( k \)-neighborhood and their estimated surface normals. By generalizing the mean curvature around a point, a multidimensional histogram is constructed to model the interactions between the directions of the estimated normals. The PFH descriptor results in a 16-D histogram descriptor for each feature point. Denote \( h_p \in R^{16} \) as the PFH descriptor for feature point \( p \). Then, the cost of matching a template feature point \( p \) to a scene feature point \( q \) is defined using \( \chi^2 \) distance [15] as follows:

\[
c(p, q) = \sum_{k=1}^{16} \frac{[h_p(k) - h_q(k)]^2}{h_p(k) + h_q(k)}. \tag{2}
\]

The feature matching costs are stored in a feature matching cost matrix \( C \in R^{n_T \times n_S} \), whose entry \( C_{ij} = c(p_i, q_j), i = 1, 2, \ldots, n_T; j = 1, 2, \ldots, n_S \) stores the cost of matching the \( i \)th template feature point to the \( j \)th scene feature point.

The matching function \( f(\cdot) \) is usually modeled as a binary variable matrix \( X \in \{0, 1\}^{n_T \times n_S} \), as follows:

\[
f(p_i) = q_j \Leftrightarrow X_{ij} = 1, i = 1, 2, \ldots, n_T; j = 1, 2, \ldots, n_S. \tag{3}
\]

\( X_{ij} = 1 \) (or zero) indicates that the \( i \)th template feature point is matched (or not matched) to the \( j \)th scene feature point. Because each template feature point is matched to exactly one scene feature point, each row of \( X \) contains exactly one entry with a value of 1.

By integrating the feature matching cost matrix \( C \) and the binary variable matrix \( X \), the feature matching cost term in (1) is expressed as

\[
\sum_{i=1}^{n_T} c(p_i, f(p_i)) = tr(CX^T) = \sum_{i=1}^{n_T} \sum_{j=1}^{n_S} C_{ij} X_{ij}. \tag{4}
\]

For the \( i \)th row of \( X \), the column index of the entry with a value of 1 specifies the index of the scene feature point matched to the template feature point \( p_i \). Let \( X_i \) denote the \( i \)th row of \( X \). Then, \( X_i S \) computes the coordinates of the matched scene feature point for \( p_i \). Combining all rows of \( X, XS \) represents the coordinates of all matched scene feature points in the same order as the template feature points.

C. Locally Affine-Invariant Geometric Constraint

Considering the affine transformations, occlusions, incompleteness, and scales, the geometric matching cost function \( g(\cdot) \) in (1) is modeled by locally affine-invariant geometric constraints. We assume that each template feature point \( p_i \) can be exactly represented by an affine combination of its neighbors as follows:

\[
p_i = \sum_{p_j \in N_{p_i}} W_{ij} p_j \tag{5}
\]

where \( N_{p_i} \) is the local neighborhood of point \( p_i \); \( W \) is a \( n_T \times n_T \) weight matrix that records the affine combination coefficients for all template feature points. Denote \( W_i \) as the \( i \)th row of \( W \). Then, \( W_i \) depicts the local geometric layout round \( p_i \). The following constraints are placed on \( W \): 1) \( W_{ij} = 0 \) if \( p_j \notin N_{p_i} \), and 2) each row must add up to one. The first constraint reflects a point’s local geometric property; the second constraint makes the representation invariant to global transformations. These two constraints guarantee that a feature point is always exactly represented by the affine combination of its neighbors [16]. By using the locally affine-invariant geometric constraints, the geometric matching cost term in (1) is defined as

\[
\sum_{i=1}^{n_T} g(p_i, N_{p_i}; f(p_i), N_f(p_i)) = \| (I - W)XS \|_1 \tag{6}
\]

where \( \| \cdot \|_1 \) denotes the summation of absolute values of all the entries in a matrix.

D. Overall Objective Function

By combining the feature and geometric matching costs in (4) and (6), we define the overall objective function for point cloud object matching as follows:

\[
\begin{align*}
\text{minimize} & \quad tr(CX^T) + \lambda \| (I - W)XS \|_1, \\
\text{subject to} & \quad X \in \{0, 1\}^{n_T \times n_S}, X1_{ns} = 1_{n_T}, XT1_{n_T} \leq 1_{n_S}. \tag{7}
\end{align*}
\]

There are three constraints in (7) as follows.

1) \( X \in \{0, 1\}^{n_T \times n_S} \) indicates a point-to-point matching pattern, where the matching between a template feature point and a scene feature point is either “Successful” \( (X_{ij} = 1) \) or “Failed” \( (X_{ij} = 0) \).

2) \( X1_{ns} = 1_{n_S} \) indicates that a template feature point exactly matches one scene feature point.

3) \( XT1_{n_T} \leq 1_{n_S} \) defines a one-to-one matching pattern, where a scene feature point is matched, at most, by one template feature point.

The problem in (7) has a nonlinear objective function with integer constraints. Such a formulation constructs an NP-hard problem and cannot be solved efficiently in polynomial time. To provide an appropriately optimal solution, we develop a Linear Programming and Simulated Annealing (LPSA) algorithm. The LPSA algorithm starts by solving the following relaxed linear programming problem in a continuous space:

\[
\begin{align*}
\text{minimize} & \quad tr(CX^T) \\
\text{subject to} & \quad X \in [0, 1]^{n_T \times n_S}, X1_{ns} = 1_{n_T}, XT1_{n_T} \leq 1_{n_S}. \tag{8}
\end{align*}
\]

Solving the linear programming problem results in a continuous solution of \( X \). To map the continuous solution to a discrete integer solution space, for each row of \( X \), we simply set the entry with the maximum value to be 1; the others are set to be zero. Then, a simulated annealing procedure, based on (7), iteratively refines the initial solution. The simulated annealing procedure takes into consideration both feature and geometric matching costs. By iteratively adjusting the mismatched points based on geometric constraints, we finally obtain an optimal matching.
The LPSA algorithm works efficiently when both the number of template and scene feature points is small. When the number of feature points increases, the size of $X$ increases dramatically. To reduce the computational complexity, we use a lower convex hull trick [17] to obtain the initial matching solution. In this way, only the scene feature points on the lower convex hull of the feature matching costs are considered as the matching candidates for each template feature point.

III. RESULTS AND DISCUSSION

The 3-D point clouds used in this study were acquired by a RIEGL VMX-450 MLS system. This system integrates two full-view RIEGL VQ-450 laser scanners that generate a maximal effective measurement rate of 1.1 million measurements per second and a line scan speed of up to 400 scans per second. The accuracy and precision of the scanned point clouds are within 8 mm and 5 mm, respectively.

A. Three-Dimensional Correspondence

Here, we evaluate the performance of the proposed 3-D object matching framework in computing 3-D one-to-one correspondences in 3-D point clouds. First, we manually selected a group of salient feature points from the template and scene point clouds, respectively. Since we aimed to find one-to-one correspondences, we set the number of template and scene feature points to be identical, i.e., $n_T = n_S$. Next, we applied the proposed object matching framework to the selected data sets to compute one-to-one correspondences between template and scene feature points. We selected the neighbors of a feature point as the $k$-nearest neighboring feature points, where $k$ was set at 5 in all of our experiments. We set the value of $\lambda$ at 1.0 to assign equal weights to the feature and geometric matching costs. Fig. 1 shows the correspondences computed for 12 groups of 3-D point clouds, each of which contains a template object and a scene object.

We also compared the proposed 3-D object matching framework with several existing methods, including 3-D shape context [18], bilateral map [19], curve analysis [20], spectral descriptor [21], and our proposed framework without local geometric constraints. As shown in Fig. 2, the matching performance of different methods is evaluated by the matching errors represented by the percentage of mismatched template feature points. Comparatively, our method outperforms other
methods and achieves a very low average matching error of 0.79%. Particularly, comparing the average matching errors obtained using our method with and without local geometric constraints (0.79% versus 51.87%), we conclude that performance is greatly improved when locally affine-invariant geometric constraints are considered.

B. Three-Dimensional Object Detection

Our 3-D object matching framework was embedded in a 3-D object detection method for detecting point cloud objects. First, the point clouds were preprocessed to remove ground points and segment off-ground points into clustered objects [2]. Next, by using the sampling method proposed in [22], \( n_T \) (\( n_T = 30 \)) feature points were evenly sampled, respectively, from 1) each clustered object and 2) the point cloud object prototype representing an instance from a specific class of objects to be detected. Then, the object matching framework was carried out to match the feature points on the prototype to the feature points on each clustered object. The value of \( \lambda \) was set at 1.0, 3.0, and 2.0 for matching light poles, palm trees, and cars, respectively. Finally, the matching costs of all clustered objects were thresholded to determine the objects of interest. By receiver operating characteristic (ROC) curve analysis, the optimal thresholds selected for detecting light poles, palm trees, and cars were set at 1.4, 3.5, and 3.2, respectively.

In this letter, three data sets of MLS point clouds were selected for evaluating 3-D object detection performance. The first data set, containing 215 million points and covering a distance of about 1742 m in the road direction, was selected to evaluate the performance of light pole detection. The second data set, containing 320 million points and covering a length of about 1813 m along the road, was used to assess palm tree detection performance. The third data set, containing 185 million points and covering a road segment of about 881 m, was chosen for the evaluation of car detection performance. Fig. 3 shows the selected point cloud object prototypes of light poles, palm trees, and cars. The ground truth and detection results for these three data sets are detailed in Table I. “Targets” denotes the number of true positives; “Non-targets” denotes the number of false positives. The proposed method was implemented using C++ and executed on an HP Z820 workstation. The computing times for these three data sets are 1924, 2412, and 1013 s, respectively. Fig. 4 presents parts of the detection results for light poles, palm trees, and cars from the three data sets. To quantitatively evaluate the detection accuracy, three measures such as completeness (\( \text{cpt} \)), correctness (\( \text{crt} \)), and quality (\( \text{qut} \)) [23] were adopted. These three measures are defined as follows: \( \text{cpt} = TP / (TP + FN) \), \( \text{crt} = TP / (TP + FP) \), and \( \text{qut} = TP / (TP + FP + FN) \), where \( TP \), \( FN \), and \( FP \) denote the number of true positives, false negatives, and false positives, respectively. The evaluation results (see Table I) demonstrate that the proposed 3-D object matching framework is very promising.

Comparative studies were also conducted to further compare the performance of the proposed method with an existing car detection method based on bottom-up and top-down descriptors [24] and a percentile-based light pole detection method [6]. The car and light pole detection results are shown in Figs. 5 and 6, respectively. As shown by the blue boxes in Figs. 5(a) and 6(a), some cars with extensive incompleteness and some light poles with advertising boards attached were not detected. However, such objects are correctly detected using our method. Thus, we conclude that our method is more effective and achieves better performance than the methods in [24] and [6].

C. Point Cloud Object Classification

As an extensive application, we integrated the object matching framework into a point cloud object classification method. The normalized cut classification algorithm [25] was used to classify point cloud objects into different categories. First, we organized the point cloud objects into a complete weighted graph, whose nodes are represented by the point cloud objects and whose edges are connected between each pair of nodes. Then, we evenly sampled \( n_T \) (\( n_T = 30 \)) feature points from each point cloud object using the sampling method proposed in [22]. The weight \( w_{ij} \) on edge \((i, j)\) was defined, using the matching cost between point cloud objects \( i \) and \( j \), as follows:

\[
 w_{ij} = \exp \left( -\left( \text{tr}(\text{CX}^T) + \lambda \| (I - W)XS \|_1 \right) \right). \tag{9}
\]
point cloud object classification. Our method achieves a classification error of 3.75%. Therefore, Fig. 8 details the ground truth and classification results.

objects from 8 classes, each of which contains 10 instances (see the point cloud objects. In this letter, we selected 80 point cloud cut classification algorithm to the weighted graph to classify geometric matching costs. Finally, we applied the normalized λ

IV. CONCLUSION

This letter has presented a framework for 3-D object matching in 3-D MLS point clouds. A locally affine-invariant geometric constraint was proposed to improve the matching performance and handle the conditions of affine transformations, occlusions, incompleteness, and scales. Each feature point in 3-D MLS point clouds is locally represented by an affine combination of its neighbors. Such a representation results in an NP-hard problem of the overall objective function. As an approximately optimal solution, an LPSA algorithm was developed to provide a polynomial time solution. The proposed 3-D object matching framework was integrated into 3-D correspondence computation, 3-D object detection, and point cloud object classification. An average matching error of 0.79% was achieved in computing 3-D correspondence; a classification error of 3.75% was obtained on point cloud object classification. Through quantitative evaluations, our proposed method achieves a completeness, correctness, and quality of over 0.96, 0.94, and 0.91, respectively, on the detection of 3-D light poles, palm trees, and cars. Comparative studies also demonstrated that our method outperforms other methods in 3-D correspondence computation and 3-D object detection from MLS point clouds.

REFERENCES


