Detection of urban features by multilevel classification of multispectral airborne LiDAR data fused with very high-resolution images

Yasmine Megahed[a,*] **Wai Yeung Yan**[a,b] **and Ahmed Shaker**[a]

[a]Ryerson University, Department of Civil Engineering, Faculty of Engineering and Architectural Science, Toronto, Canada

[b]Hong Kong Polytechnic University, Department of Land Surveying and Geo-Informatics, Faculty of Construction and Environment, Hung Hom, Kowloon, Hong Kong

Abstract. A complex pattern of urban demographic transition has been taking shape since the onset of the COVID-19 pandemic. The long-standing rural-to-urban route of population migration that has propelled waves of massive urbanization over the decades is increasingly being juxtaposed with a reverse movement, as the pandemic drives urban dwellers to suburban communities. The changing dynamics of the flow of residents to and from urban areas underscore the necessity of comprehensive urban land-use mapping for urban planning/management/assessment. These maps are essential for anticipating the rapidly evolving demands of the urban populace and mitigating the environmental and social consequences of uncontrolled urban expansion. The integration of light detection and ranging (LiDAR) and imagery data provides an opportunity for urban planning projects to take advantage of its complementary geometric and radiometric characteristics, respectively, with a potential increase in urban mapping accuracies. We enhance the color-based segmentation algorithm for object-based classification of multispectral LiDAR point clouds fused with very high-resolution imagery data acquired for a residential urban study area. We propose a multilevel classification using multilayer perceptron neural networks through vectors of geometric and spectral features structured in different classification scenarios. After an investigation of all classification scenarios, the proposed method achieves an overall mapping accuracy exceeding 98%, combining the original and calculated feature vectors and their output space projected by principal components analysis. This combination also eliminates some misclassifications among classes. We used splits of training, validation, and testing subsets and the *k*-fold cross-validation to quantitatively assess the classification scenarios. The proposed work improves the color-based segmentation algorithm to fit object-based classification applications and examines multiple classification scenarios. The presented scenarios prove superiority in developing urban mapping accuracies. The various feature spaces suggest the best urban mapping applications based on the available characteristics of the obtained data. [DOI: 10.1117/1.JRS.15.044521]

Keywords: aerial images; artificial neural networks; color-based segmentation; data integration; multilevel classification; multispectral airborne LiDAR; principal component analysis; spectral–geometric features; urban mapping.

Paper 210622 received Sep. 24, 2021; accepted for publication Dec. 15, 2021; published online Dec. 30, 2021.

1 Introduction

The United Nations' world urbanization prospects of 2019 anticipate the world urban population to increase to 4.9 billion versus the world rural population dropping by ∼28 million between 2005 and 2030. Meanwhile, more than half the world's population resides in metropolitan cities, and the proportion is projected to reach two-thirds by 2050.[1]

Urban sprawl has significant ecological, social, and health ramifications. Of the considerable negative impacts that urban sprawl brings to the ecosystem, the loss of agricultural lands, air

*Address all correspondence to Yasmine Megahed, yasmine.megahed@ryerson.ca

pollution, and deterioration of water resources are most fateful to the long-term sustainability of humanity. On the societal level, uncontrolled urban expansion causes enormous strains on social institutions. Local governments face the dilemma of increasing public spending or risk aggravating existing health and social issues, such as poverty, crime, obesity, unemployment, and social isolation.[2]

The 2018 revision of the United Nations' world urbanization prospects[3] addressed North America among the most urbanized geographic regions, with 82% of its population living in urban areas in 2018. Urban residents in North America are more than doubled in 1950 to 2018 (110 versus 299 million) and are expected to increase by 29% in 2050. The expansion in urban dwellers results from industrialization and rapid economic growth, which offer opportunities in education and employment. In contrast, the urban population growth rate declined from 1995 to 2000 to 2015 to 2020 (1.6% versus 1%) and is projected to further decrease to 0.6% in 2045 to 2050. Nevertheless, the still-positive growth rate exhibits North America's high levels of urbanization.

The COVID-19 pandemic brings to the fore some of the fragilities of urbanization. The high-density living, patterns of human contacts, and the underlying social issues in many urban areas provide an easy transmission route for the novel coronavirus. Despite only 56% of the world population being urban, 95% of the COVID-19 cases occur in urban settlements, with more than 1500 metropolitan cities economically impacted. According to the World Bank,[4] 49 million urban residents worldwide are threatened by pandemic-driven new poverty. Moreover, the contagions and the COVID-19 lockdowns imposed by municipal authorities have affected the dynamics of urban–urban and rural–urban migrations.[4] Remote work has encouraged urban dwellers, especially long commuters, to purchase larger suburban homes offered at lower interest rates.[5] For instance, Toronto residents spent on average 16.1 h daily indoors at home before the pandemic hit.[6] This average is expected to have increased during the COVID-19 pandemic, with working from home and the absence of outdoor activities being the norm. Hence, urban inhabitants have been seeking extra spaces to accommodate simultaneous work and school, where they can better practice hygiene and physical distancing without risking contagion in apartment buildings' related facilities (i.e., elevators and lobbies).[7]

These dynamics necessitate policymakers, demographic researchers, and nongovernmental organizations to improve current plans and develop new strategies to better manage hotspots of anticipated service shortages due to urban sprawl. Urban mapping represents a scientific guidance to municipal leaders that considers spatial factors with other urban-related datasets for more precise urban assessments and predictions.

Light Detection and Ranging (LiDAR) or laser scanning technology collects high-quality three-dimensional (3D) data about topographic objects on the Earth's surface. Airborne LiDAR systems acquire highly accurate, shadow-free, georeferenced, and unstructured distributed 3D point clouds with minimum point spacing. In addition, some airborne laser scanners have multiple-return and full-waveform privileges. The multiple-return characteristic allows each emitted laser signal to sense multiple objects and collect the backscattered laser signal strength together with the recorded time in a waveform data structure, providing a more comprehensive understanding of the targets' physical characteristics.[8] Current multispectral airborne LiDAR scanners sense objects using a maximum of three laser channels (i.e., Optech Titan). This coarse spectral resolution encourages fusing LiDAR data with high-resolution multispectral images to obtain a richer inventory of spectral and textural information.[9] In this way, LiDAR imagery fusion combines the advantage of reflectivity variation of several wavelength ranges on the electromagnetic spectrum (i.e., visible red, green, and blue (RGB) and near-infrared (NIR)], with the geometric data description in the LiDAR's 3D domain. Consequently, the classification of LiDAR point clouds obtained for urban regions could be achieved with better mapping results.

Huang et al.[10] divided LiDAR point clouds using voxelization into chips of points. They downsampled the size of the chips to represent their main structure. Instead of using traditional calculated feature spaces, they input the downsampled chips to PointNet++ to learn the points' features. PointNet++ is a deep learning technique that divides input data into overlapping subdivisions, learns the local features of each subdivision, successively groups lower-level local features to learn higher-level features until the learning of global features of the entire input data is achieved. PointNet++ outputs initially classified data with soft labels. The authors

constructed a weighted graph for global regularization that accounts for the initial label probability set and the spatial correlation to refine the soft labels. They achieved 85.38% overall accuracy, with 70%, 79%, 97%, 6%, 89%, and 89% accuracy of identifying man-made terrains, natural terrains, high vegetations, low vegetations, buildings, and vehicles, respectively.

Sen et al.[11] carried out an unsupervised classification on airborne LiDAR point clouds acquired for a residential urban area using the weighted self-organizing maps clustering technique. They applied Pearson's chi-squared independence test to weigh the normalized data attributes, 3D coordinates, and a single intensity. They manually adjusted the number of clusters based on the visual observation of the resulting clusters and labeled them manually using 3D visual analysis and satellite images. The authors employed Cramer's V coefficient to define the strength of association between the LiDAR data's attributes and output clusters. They reached a mapping accuracy of 86% and per-class accuracies of 93%, 62%, 74%, and 96% for buildings, vegetations, transmission lines, and ground, respectively.

Kang et al.[12] achieved a higher overall accuracy of 95% by integrating airborne LiDAR point clouds and RGB aerial images. They used the orthoimages' direct georeferencing data to register both data types. The authors applied the k-nearest neighbor (k-NN) and fractal net evolution approach-based algorithms to segment LiDAR and imagery data before spectral and geometrical feature extraction. They introduced an improved mutual information-based Bayesian network structure learning algorithm for data classification at multiple neighborhood and segmentation scale sizes. They compared the results with Dtree, AdaBoost, random forest (RF), and support vector machines (SVM) classifiers. The authors recommended their proposed Bayesian network for ground, low vegetation, and high vegetation over buildings' land-uses. They obtained an accuracy of 96%, 93%, 97%, and 90% for the four classes, respectively.

Similarly, Sanlang et al.[13] fused airborne LiDAR point clouds with high-resolution aerial images obtained with RGB and NID bands. They segmented the images by eCognition software to avoid the salt-and-pepper effect commonly encountered in land-cover classification. However, the authors introduced different spectral, textural, geometrical, and 3D urban structural parameters and applied the Gini Index to measure the significance of each extracted feature. They followed a multimachine learning approach using the RF, k-NN, and linear discriminant analysis to classify an urban scene with buildings, trees, grass, soil, impervious ground, and water. Their findings included a 3% increase in overall accuracy when considering the LiDAR's 3D geometric characteristics in the feature space. They also concluded that the digital surface model (DSM) was the most critical feature. Nevertheless, the reported overall accuracy barely passed 87% with the RF classifier, and the maximum class accuracy did not exceed 93%.

In a trial to target vegetations around urban settlements for fire reduction and controlling plans, Rodríguez-Puerta et al.[14] classified no-vegetation, crops, bush and grass, permitted and forbidden trees using RF, linear and radial SVM, and artificial neural networks (ANNs). They introduced nine data combinations derived from high-density unmanned aerial vehicle LiDAR, low-density airborne laser scanning LiDAR, Pleiades-1B, and Sentinel-2 data. The satellites acquired the images in RGB and NIR bands, with an additional short-wave infrared band for Pleiades-1B photos. The authors used the bands to calculate vegetation indices and growth metrics. They applied the variable selection using RF to select the final classifying variables based on the Gini Index. Similar to other related research, the authors used the eCognition software for the multiscale segmentation of the imagery data. The best overall mapping accuracy they could achieve was 92% by classifying the Sentinel-2 fused by both LiDAR data using the RF classifier.

Likewise, Pu and Landry[15] combined multiseasonal Pleiades images with airborne LiDAR point clouds to map seven urban tree species. The authors computed spectral and geometric features from the imagery objects and transformed them into fewer canonical variables. They added normalized DSM-derived variables and introduced seasonal trajectory difference indices for two-seasonal combined images. They carried out a multilevel classification using RF and SVM. Their expanded feature space reached an overall accuracy of 75% using the SVM classifier. Zhang and Shao[16] also mapped urban vegetation, but into forest and grassland classes by combining airborne LiDAR point clouds and multispectral Worldview-2, Worldview-3, and GaoFen-2 satellite images. They developed canopy- and band-related features to five classification models: stepwise linear regression, k-NN, ANN, support vector regression, and RF.

Of the five models, the RF produced the highest coefficient of determination ($R^2 \approx 0.70$) and the lowest root-mean-square and relative-root-mean-square errors.

Urban sprawl, usually associated with accelerated economic expansions in developed countries, results in large-scale reclamation projects with intense constructions that compact soils with land subsidence and building collapse threats. He et al.[17] integrated airborne LiDAR point clouds with interferometric synthetic aperture RADAR (InSAR) imagery data for producing urban subsidence hazard maps. They calculated land subsidence rate information using small baseline subset (SBAS)-InSAR and permanent scatters (PS)-SBAS-InSAR algorithms on multitemporal Sentinel-1A and TerraSAR-X images, respectively. After removing distorted RADAR scatters due to shadow, layover, and foreshortening effects, the authors used fine-resolution DSM data derived from airborne LiDAR scanners for a further precise geometric correction of SAR images. They classified the DSM for building extraction and then applied a feature combination method to extract contour lines. They considered building heights and building contour lines as driving forces in assessing buildings' subsidence hazard levels.

Our study tests the hypothesis of expanding LiDAR point clouds' feature space by fusion with imagery data to enhance urban mapping accuracies. Specifically, we aim to (i) enhance the color-based segmentation technique[18] to fit supervised object-based classification of colored airborne LiDAR point clouds after integration with aerial photos and (ii) introduce a detailed multilevel classification of LiDAR data using 10 feature spaces formed from different combinations of variables based on the multispectral properties of LiDAR-imagery data and the 3D geometric characteristics of LiDAR point clouds. Accumulating multispectral LiDAR-imagery data's original and derived features helps improve mapping accuracies, eliminate misclassifications, and set a reference for matching potential urban mapping applications based on the available properties of the data under processing, which marks the study as superior to other related research work.

The remaining sections address methods in Sec. 2, experimental work in Sec. 3, results and discussions in Sec. 4, conclusions in Sec. 5, and Appendix in Sec. 6 of confusion matrixes on the testing data for the entire introduced classification scenarios (feature combinations).

2 Methods

Figure 1 schematically explains the conceptual overview of the proposed methodology. First, LiDAR point clouds are georegistered to imagery data covering the same area of interest; consequently, the spectral properties of the imagery data densify the LiDAR data's feature space. Then, LiDAR data are geometrically classified based on height to separate ground from nonground points. We recommend ground filtering to avoid misclassification of objects sharing similar spectral characteristics (i.e., grass and trees). Afterward, the 3D point clouds are segmented, then different radiometric and geometric features are calculated for each segment. Later, segments are collected for classification models' training, validation, and testing. Next, an object-based classification runs on the LiDAR point cloud to test different feature combinations on the mapping accuracy. One of these scenarios is lessening an extended feature space to an optimal combination without significantly affecting the classification accuracy. All classification scenarios are evaluated for comparison and final land-use map production.

2.1 *LiDAR-Image Georegistration*

The first step of the fusion process is to perform georegistration between LiDAR and imagery data. We georegistered multispectral LiDAR point clouds to an aerial photo using a phase congruency (PC)-based scene abstraction approach.[19] LiDAR 3D points are converted to two-dimensional (2D)-intensity or height images, whichever type better describes the scene based on the visual interpretation. For example, an area of a wastewater treatment plant varies in height more than in intensity, where most elements are inner and outer tanks, curbs, and structures. These elements share the same concrete cover; and thus, a height-based interpolation would be the optimal decision. On the other hand, a residential area typically includes urban features varying in intensity (i.e., grass, asphalt, and land-markings), which advises an interpolation based on LiDAR data's intensity values. The approach implements the PC filter, which computes

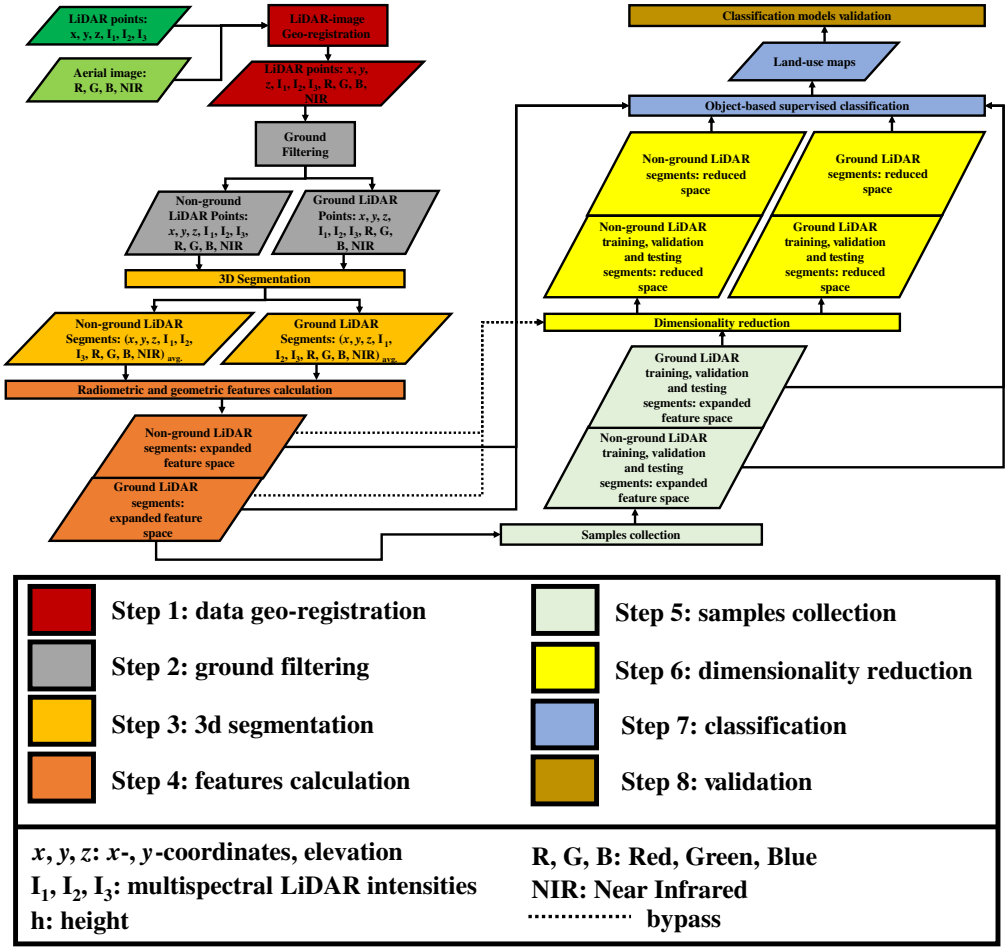


Fig. 1 Proposed methodology of multilevel classification of LiDAR point clouds fused with very high-resolution images.

the moment of each pixel's center point, knowing its PC measure in different orientations. The moment value of a point indicates whether it is an edge or a corner point. A predefined threshold range of moment values can be set to identify candidate tie points on both data sets.

Georegistering data acquired at different times with different sensors may result in two dissimilar sets of candidate tie points, impacting the threshold range. In this case, the PC filter's outputs (moment images) are abstracted by clustering, then detecting common polylines in LiDAR and imagery data. Moment points within a buffer around these detected edges are considered the candidate tie points. Alternatively, an additional filter can be fused with the PC filter to isolate candidate tie points inputted to the shape context descriptor model to be matched in pairs of final tie points. Finally, a least-squares adjustment estimates the transformation parameters of empirical registration models.

This registration is generic, as it was found to accommodate different urban morphologies, is no longer limited to traditional linear control primitives, and does not require the onboard acquisition of both data simultaneously.[19]

2.2 *Color-Based Segmentation*

After data registration, LiDAR point clouds are expressed by 3D coordinates as well as their spectral characteristics. The spectral characteristics include the ones originally captured by a LiDAR sensor during the same flight mission and those inherited from aerial images. The color-based segmentation algorithm proposed by Zhan et al.[18] is applied to segment LiDAR point clouds based on their geometric and spectral characteristics. The approach determines the

similarity between two points by calculating the geometric and colorimetric distances between one another. It measures colorimetric distances based on the RGB signatures of LiDAR points. It has been applied successfully in previous research work in segmentation-oriented applications.[20,21] The algorithm performs the color-based segmentation in two steps: segment growing and segment merging and refinement.

2.2.1 *Segment growing*

In this step, the algorithm assigns each unlabeled LiDAR point to a segment and then marks it as labeled. It constructs three entities: points (P) that contains the LiDAR points to be segmented, stack (ST), and segments (S). In the case of an empty ST, which is the default setting when the algorithm starts, the process loops on P until it meets an unlabeled point (p). The process appends p to ST, creates a new segment (s) in S, inserts p to s, and eventually marks p labeled in P. While ST is occupied, it pushes its top point, point of investigation (pt) out, and the process searches its neighboring points in P within a 3D distance window. If the neighbors are unlabeled and also radiometrically close to pt, the process appends them to s and ST. Once ST is clear, densifying s with LiDAR points terminates, and the algorithm looks for another unlabeled point in P to initiate a different segment in S. The process continues until ST is empty and all points in P are labeled.

Figure 2 addresses a hypothetical example to illustrate the segment growing step in the color-based segmentation process. Figure 2(a) represents the start run, where all points P are not assigned to segments and yet marked unlabeled, and ST and S are created. ST is by default empty. Hence, the process searches for an unlabeled point (p_1) in P, appends it to ST, constructs a new segment s_1 in S, and adds p_1 to s_1. ST's pt is p_1 that becomes the point of investigation [Fig. 2(b)]. The algorithm marks p_1 labeled in P. ST pushes out its pt (p_1), and the process locates its neighbors (p_2, p_3, and p_4) in P within a predefined 3D distance range [Fig. 2(c)]. The three neighbors are unlabeled, but only p_2 meets the radiometric similarity condition. Thus, the algorithm appends p_2 to s_1 and ST and marks it labeled in P [Fig. 2(d)]. Consequently, the process does not add p_2 to a different segment in the future to maintain a one-to-one relation between P and S. Since ST is occupied, points inside can contribute to s_1's growth by their neighbor points as long as they are close in color. ST pushes out its pt (p_2), which turns to be the point of investigation. The approach determines p_2's neighbors in P (p_1, p_3, p_4, and p_5). The latter three points are unlabeled, but none of them meets the colorimetric condition. Therefore, the densification of s_1 always ends with an empty ST [Fig. 2(e)].

The process loops again on P and locates the following unlabeled point (p_3), which initiates a newly created segment s_2, joins ST, and becomes the point of investigation [Fig. 2(f)]. The algorithm marks p_3 as labeled and finds its neighbors in P (p_1, p_2, p_4, and p_5). The latter two points are unlabeled; hence, they are the ones for the algorithm to evaluate the colorimetric condition for inclusion into s_2 [Fig. 2(g)].

2.2.2 *Segment merging and refinement*

The output of the segment growing step is S, which contains roughly segmented clusters that need to be merged and refined in this subsequent run. The algorithm builds a merged segments (MS) entity to store lists of homogeneous segments from S. The process marks S's segments as labeled if MS already includes those segments. Otherwise, they are marked as unlabeled. The approach first iterates S to find an unlabeled segment (s). Then, a new merged segment (ms) is created in MS, with s being a member of ms. Afterward, s is marked as labeled and becomes a segment of the investigation. The process locates its neighboring segments in S within a 3D searching window. s's neighbors are determined by locating the point neighbors of all points within s. Every segment to which these point neighbors belong is a neighboring segment. The algorithm calculates the radiometric similarity between s and its unlabeled neighbor segments after averaging the RGB values of each. Suppose that s's neighboring segments are found to be colorimetrically close to s, the algorithm appends them to ms and marks them labeled in S. The process then continues looping on S and merging segments in MS and terminates when all segments in S are marked labeled and included in MS. Finally, all segments within the same MS are

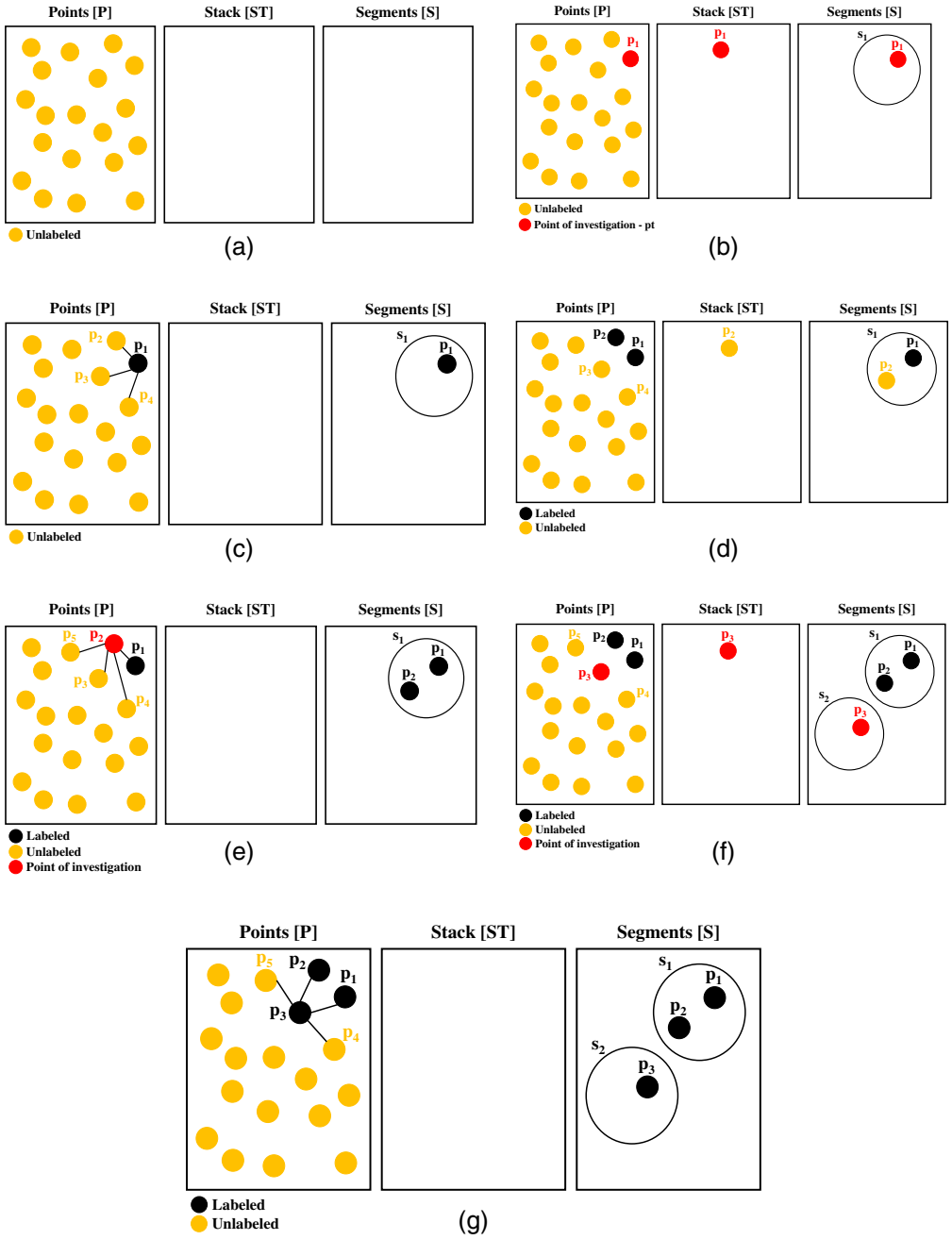


Fig. 2 Color-based segmentation algorithm applied to LiDAR point clouds: segment growing step. (a) P, ST, S are created, and ST is empty. (b) Unlabeled p_1 in P initiates s_1 in S, is pushed to ST, and becomes ST's pt and point of investigation. (c) p_1 is marked labeled in P, pushed out by ST, and its neighbors are determined in P. (d) p_2 meets the conditions, is appended to s_1, is pushed to ST, becomes ST's pt, and is marked labeled in P. (e) ST pushes out p_2 that becomes the point of investigation, and its neighbors determined in P. (f) None meets the conditions, s_1 is terminated, ST becomes empty, the following unlabeled point in P is p_3 that initiates s_2 in S, p_3 is pushed to ST, and becomes ST's pt and point of investigation. (g) p_3 is marked labeled in P, pushed out by ST, its neighbors are determined in P, qualified neighbors to be examined.

fused. The refinement step looks for MSs with a number of LiDAR points less than a predefined minimum. The process highlights them as MSs of interest, determines their nearest MS in MS within a specific 3D window size, and fuses both in a refined segment (rfs). The refinement continues until the size of all rfs is larger than the predefined minimum. That is when MS eventually turns to be an rfs entity.

Figure 3 graphically explains the segment merging and refinement step. The figure describes a hypothetical scenario where S has five unlabeled segments, and MS is empty before running the step [Fig. 3(a)]. The process loops on S, spots an unlabeled segment (s_1), marks it labeled in S, creates a new MS (ms_1) in MS, and inserts s_1 in ms_1 [Fig. 3(b)]. s_1 becomes the segment of investigation, for which the process locates its neighbors in S (s_2 and s_3) [Fig. 3(c)]. Both are unlabeled and meet the colorimetric similarity condition with s; hence, the algorithm marks them labeled in S and appends them to ms_1 in MS [Fig. 3(d)].

The process loops again on S, finds the subsequent unlabeled segment (s_4), highlights s_4 as labeled in S, and initiates (ms_2) in MS with s_4 [Fig. 3(e)]. s_4 is the segment of investigation, and the process searches its neighbors in S (s_2 and s_5) [Fig. 3(f)]. Only s_5 is unlabeled and meets the radiometric similarity condition with s_4. Therefore, the algorithm marks s_5 labeled in S and pushes it to ms_2 in MS [Fig. 3(g)]. The merging process terminates, since S's elements are entirely labeled and fully included in MS. Segments within same MSs are fused [Fig. 3(h)]. Finally, ms_2's size is less than the minimum, so it becomes an MS of investigation whose nearest neighbor (NN) in MS is ms_1 [Fig. 3(i)]. Eventually, the process merges ms_2 to ms_1 as a new refined segment (rfs_1) in RSF [Fig. 3(j)].

2.3 *Eigenvalue-Based Geometric Features Determination*

Sanderson graphically explains the geometric conception of eigenvectors and eigenvalues with a series of visual-aided materials on their website.[22,23] An eigenvector of a linear transformation is a nonzero vector on which the only effect of the linear transformation is scaling by a constant number. The value of the scaling constant is the eigenvalue of the eigenvector. Equation (1) describes the above relation as follows:

$$A\vec{v} = \lambda\vec{v}, \tag{1}$$

where A is the transformation matrix that scales its eigenvector $\vec{v}$ by an eigenvalue λ.

To solve for λ and $\vec{v}$, Eq. (1) can be expressed as

$$(A - \lambda I)\,\vec{v} = \vec{0}, \tag{2}$$

where I is the identity matrix.

Since the right-hand side of Eq. (2) is a zero-vector, $(A - \lambda I)$ is a singular transformation with the property

$$\det(A - \lambda I) = 0, \tag{3}$$

where $\det(A - \lambda I)$ is the determinant of $(A - \lambda I)$.

Equation (3) solves the eigenvalue(s) of the linear transformation A. Finally, the corresponding eigenvector of each eigenvalue can be determined using Eq. (2) upon substitution of λ with each solved eigenvalue.

Figure 4 shows a numerical example to geometrically describe how eigenvectors and their corresponding eigenvalues are calculated for a transformation matrix $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ between 2D spaces. Figure 4(a) assumes the coordinates of the unit vectors $\hat{i}$ and $\hat{j}$ in the input space are (1, 0) and (0, 1), respectively. Hence, (A) transforms both to (3, 0) and (1, 2), respectively, which represent the coordinates of the output space's base vectors with respect to the input space's grid shown in the background [Fig. 4(b)]. The procedure as mentioned earlier leads to a quadratic polynomial in λ that has two solutions: $\lambda_1 = 2$ and $\lambda_2 = 3$, meaning that there are two vectors in the input space that are only scaled by constants upon the transformation. Substituting in Eq. (2) with both eigenvalues determines the corresponding eigenvectors as expressed by $x + y = 0$ and $y = 0$, respectively [Fig. 4(c)]. A vector that lies on either eigenaxis in the input space does not alter that axis in the output space; the vector is just scaled by its corresponding eigenvalue (2 or 3) as shown in Fig. 4(d).

The geometry of LiDAR points in 3D is derived from the variance of their coordinates: x, y, and z. One way to analyze LiDAR points is to perform a coordinate transformation such that the transformed coordinates X, Y, Z optimally reveal the variances of the original coordinates.

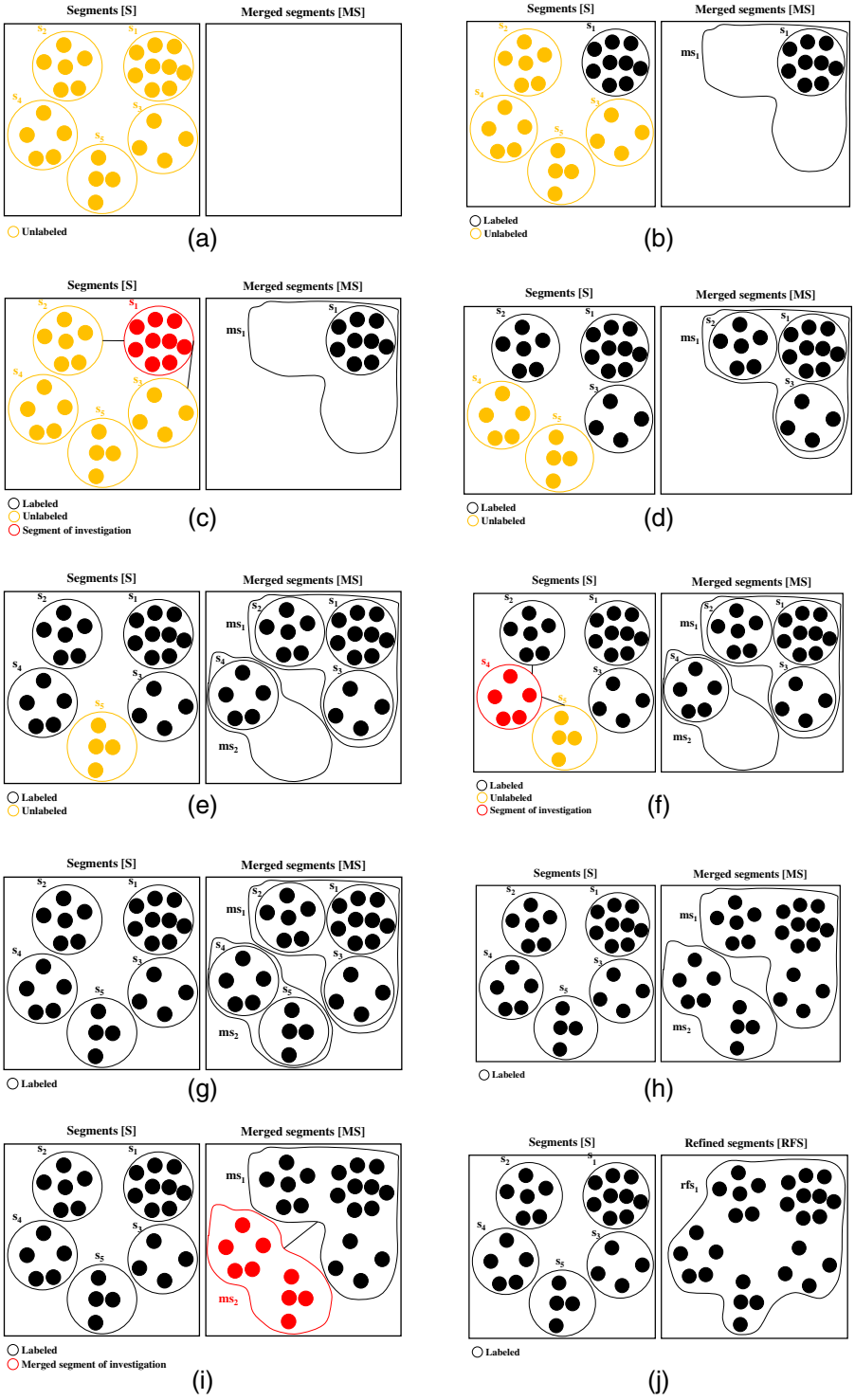


Fig. 3 Color-based segmentation algorithm applied to LiDAR point clouds: segment merging and refinement step. (a) S is from segment growing step, and MS is created. (b) Unlabeled s_1 is marked labeled in S, ms_1 is created in MS, and s_1 is inserted in ms_1. (c) Segment of investigation is s_1, and its neighbors are determined in S. (d) s_2 and s_3 meet the conditions, are marked labeled in S, and appended to ms_1 in MS. (e) Subsequent unlabeled segment is s_4, is marked labeled in S, ms_2 is created in MS, and s_4 is inserted in ms_2. (f) Segment of investigation is s_4, its neighbors are determined in S. (g) s_5 meets the conditions, is marked labeled in S, and appended to ms_2 in MS. (h) Segments are labeled in S, merging process terminates, and segments are fused within same ms in MS. (i) ms_2's size is less than the minimum, it becomes an MS of investigation, its NN (ms_1) is determined in MS. (j) ms_2 is merged to ms_1 to form rfs_1 in RSF.

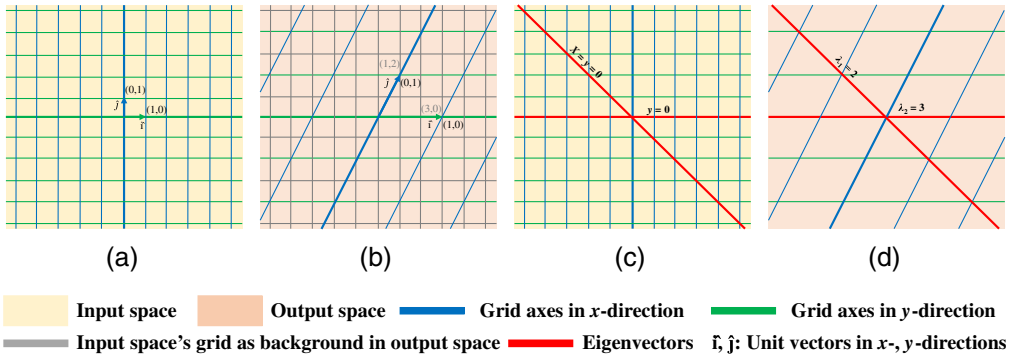


Fig. 4 Geometric interpretation of eigenvectors and eigenvalues in 2D spaces–numerical example. (a) Coordinate system of input space. (b) Coordinate system of output space. (c) Eigenvectors on input space. (d) Eigenvectors and corresponding eigenvalues on output space.

This is achieved when the covariance matrix expressing the variance/covariance between the 3D coordinate records is diagonal, meaning that the diagonal values are the variances of the transformed coordinates X, Y, and Z that have a zero or minimal correlation between each other. The diagonal covariance matrix in the transformed space (X, Y, Z) can be obtained by eigendecomposition of the original covariance matrix, with each dimension and scaling of the transformed space being an eigenvector and its corresponding eigenvalue of the original covariance matrix. Moreover, the resulted eigenvectors are always orthogonal due to the orthogonality of the transformation (covariance) matrix itself, which is a rotation matrix in this case. The corresponding eigenvalues form indices that better expose the geometry of LiDAR point clouds in a 3D space.[24,25]

2.4 *Dimensionality Reduction of Feature Space*

It is preferable to diminish the set of input variables (features) of the data being analyzed while developing a predictive model to decrease the computational cost.[26] In some cases where the space volume is too large, the data records may not be representative, causing fitting problems that reduce the model performance, in a phenomenon known as the curse of dimensionality.[27] Feature selection techniques can be supervised by eliminating statistically irrelevant variables with a weak relationship to the target variable the model attempts to predict. On the other hand, unsupervised feature selection methods drop redundant variables based on statistical measures (i.e., correlation), independent of the target variable.

Even though feature selection and dimensionality reduction techniques attempt a fewer input space to a predictive model, the latter fundamentally differs from the former. Dimensionality reduction methods project the data into a new space of fewer dimensions, resulting in transformed input features[26] which are called components in the principal component analysis (PCA) method that we applied in our study to reduce the data dimensionality.

The PCA linearly projects a feature space into a subspace that still preserves the essence of the original data.[28] It looks for descriptive features of high variance values revealed by the eigendecomposition of the features covariance matrix. Then, it projects the input feature space into another space constructed by that chosen subset of features. Brownlee[28] delineates the process in the following steps:

1. Centering the feature values by subtracting the mean.
2. Calculating the covariance matrix of the centered values.
3. Calculating the eigendecomposition of the covariance matrix. The eigenvectors picture the directions or components of the projection output space. The eigenvalues are the variances of these components (termed features in the input space) or the magnitudes for the directions.
4. Selecting the principal components (PCs) of the new subspace by ranking the eigenvectors by their corresponding eigenvalues in decreasing order and choosing those with the largest eigenvalues.
5. Projecting the input feature space into the subspace.

Table 1 Description of performed classification.

Perspective	Type	Explanation
Learning type	Supervised	Training samples of observed classes are a prerequisite for the classifier to determine the class of each segment
Data distribution assumption	Nonparametric	MLP does not assume the data probability distribution's shape. It uses kernel density estimation to approximate the probability density function of a continuous random variable
Data structure	Object-based	After LiDAR data segmentation, MLP forms fundamental knowledge of data by being trained on the mean feature values of each segment instead of the direct feature values of each point. Hence, the class MLP assigns to a segment is given to all points included within that segment
Output classes per segment	Hard	MLP assigns each segment to every class with varying probabilities. In this study, the class MLP labels a segment with is the one with the highest probability

2.5 *Classification of LiDAR Data*

We isolated a portion of the segmented LiDAR data and divided it into training, validation, and testing subsets before employing the multilayer perceptron (MLP) neural network classifier. Table 1 summarizes the main characteristics of the classification we carried out in this work.

ANNs are machine learning algorithms that are inspired by observations of how the brain functions with its constituent structures.[29] An MLP neural network consists of an input layer and some hidden layers. The last layer is the output layer that provides the final predictions. Each layer is a row of neurons (nodes), where each neuron has weighted inputs, bias, and output to the next layer, representing a perceptron.[30] An optimization algorithm weights a neuron's inputs, and a bias is accumulated to the weighted summation of the inputs to determine their signal strength.[31] The ultimate goal is setting those weights and biases to particular values so the overall classification error is minimal.[32] An activation function is applied to an output signal strength by intensifying or diminishing its value depending on its magnitude. Consequently, outputs of large magnitudes propagate further and contribute to the final predictions more than those of lower magnitudes.[32]

Training data have to be numerical in a multiclass classification application. The dimension of the input layer is set to the feature space of the training data, and the output layer has as many nodes as the number of noticed classes. The optimization algorithm initially assigns random weights to each node's inputs, and their signal strengths are determined after adding a bias to the inputs' weighted summation.[30] The activation function filters signal strengths so only those of high magnitudes propagate to the final predictions. A loss (cost) function measures the model's classification error, represented by comparing the predictions to their corresponding ground truth data. The model training process iterates over the training dataset for a preset number of times (epochs), or until a condition that signifies a cost function's minimum is numerically achieved.[31] The model updates the weights at each epoch, where all training records participate. In addition, internal weights updates occur at each batch or subset of the training data in the case of batch training.[33]

3 Experimental Work

Python programming language ran sequential calculations on Spyder Integrated Development Environment (IDE) v 3.7.9, embedded in Anaconda Enterprise v 4.10.0. ERDAS IMAGINE v 2018 helped visualize LiDAR point clouds, and LAStools converted LiDAR data into different formats and extracted metadata. Data analysis was carried out on a workstation with the following specifications: Windows 10 Pro for workstations OS 64-bit, 3.2 GHz processor (16 CPUs), and memory of 131,072 MB RAM.

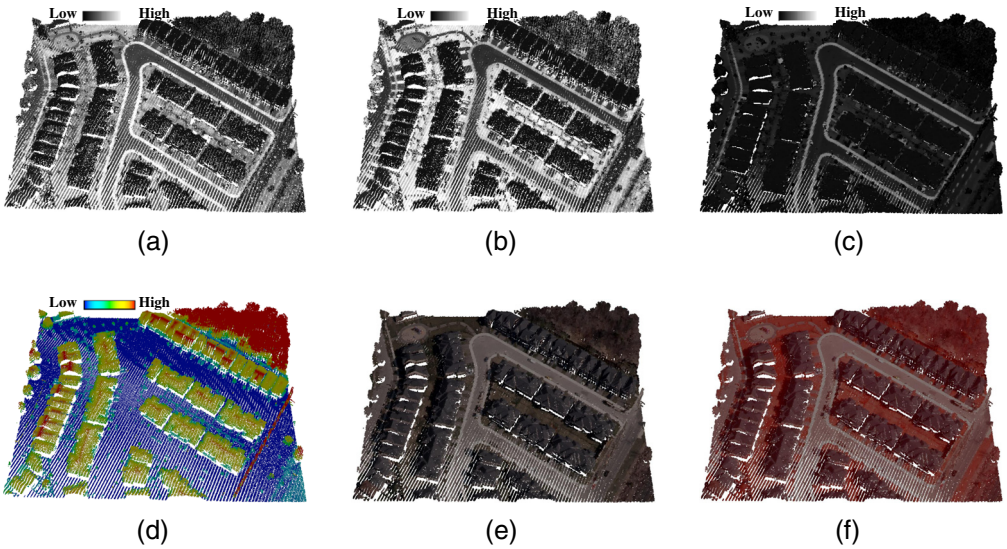


Fig. 5 Visualization of LiDAR data after registration to multispectral very-high-resolution aerial image. (a) Intensity–C_1 channel. (b) Intensity–C_2 channel. (c) Intensity–C_3 channel. (d) Height. (e) R, G, B bands visualized in RGB colors. (f) NIR, B, and G bands visualized in RGB colors.

3.1 *Study Area and LiDAR Dataset*

We used a multispectral LiDAR point cloud captured by the airborne Optech Titan sensor in 2015 to test the proposed approach. The sensor collected LiDAR data using three laser channels: C_1, C_2, and C_3 that represent the 532-, 1064-, and 1550-nm wavelengths of the electromagnetic spectrum, respectively. The point cloud contains 1,976,164 3D points spaced at 0.13 m, produced and projected by the OptechLMS software to the North American Datum (NAD) 1983 Universal Transverse Mercator (UTM) coordinate system (zone 17N). The dataset covers a 33,000 m^2 residential area in Rouge within Scarborough, east of Toronto. Megahed et al.[34] georegistered the points to a very high-resolution orthophoto acquired in 2014 with a spatial resolution of 0.2 m and covers the same study zone in R, G, B, and NIR bands. They later corrected the overparameterization problem in empirical registration models.[35] Figure 5 visualizes the LiDAR dataset before and after the georegistration.

We used the "lasground-new" tool[36] within the collection of LAStools software package to separate ground from nonground points. The tool applies the progressive triangulated irregular network (TIN) densification approach.[37] It filtered the LiDAR data into 857,738 and 1,118,426 ground and nonground points, respectively.

We observed four nonground and another four ground classes, as follows:

1. buildings,
2. vehicles,
3. high vegetation (tall trees),
4. low vegetation (short trees, shrubs, and bushes),
5. dark asphalt (collectors),
6. light asphalt (local roads),
7. sidewalks, and
8. grass.

3.2 *Color-Based Segmentation of LiDAR Data*

Figure 6 schematically explains how the segmentation algorithm was applied in the study. It summarizes what Secs. 2.2.1 and 2.2.2 explain in a single chart, with a few alternations that are further illustrated at the end of this section. The classic k-NN method is commonly utilized to determine a point's neighbors. It requires a predetermined k value, representing a chosen number of neighbors

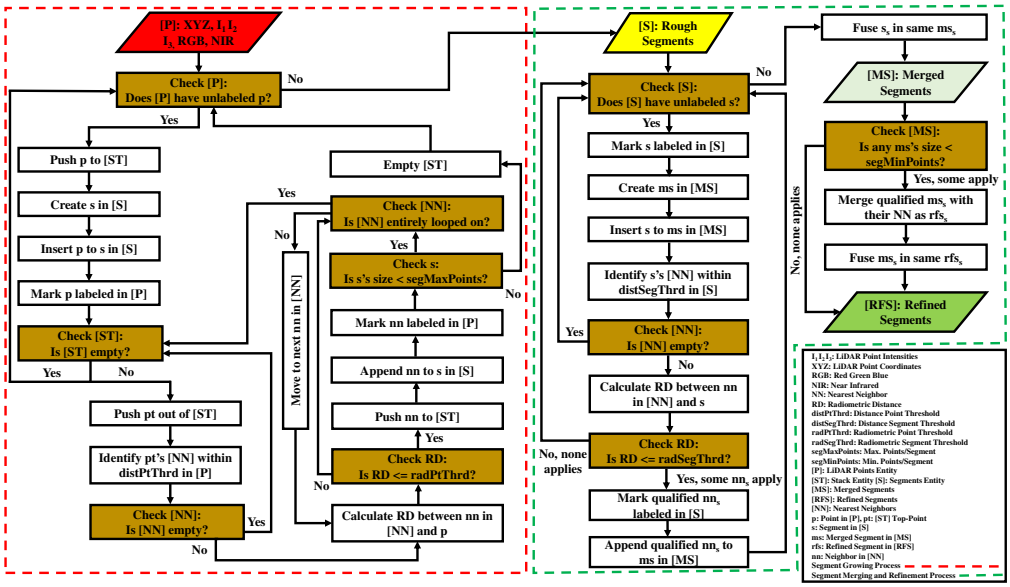


Fig. 6 Application of color-based segmentation algorithm with some modifications.

for the query point, to initialize the process. First, the algorithm calculates the distances between the point and the remaining data samples. Then, it creates a collection where the indices of those samples are stored with their distances from the query point, sorted in ascending order. Finally, the method selects the first k records from the constructed collection as the point's k-NN.[38] Despite the simplicity of the k-NN, such a brute-force search is structureless; consequently it is computationally expensive when processing multidimensional data with large sizes.[38,39] Hence, the structure-based k-dimensional tree (k-d tree) method was applied in this study.

A k-d tree is a binary tree where each nonterminal node divides the data into two portions depending on a record's position from a k-d hyperplane (splitting partition).[40] Each nonterminal node depicts a different partition. Each level of nonterminal nodes alternates sequentially among the d dimensions in splitting the records. The search starts at the root node and goes down the tree, turning left or right at each nonterminal node based on the query point's value compared with the threshold value at the split dimension.[41] The search continues until it reaches the terminal node that contains a maximum number of points at which the algorithm switches over to brute-force (leaf size).[42] However, these points do not represent the final set of NNs for the query point if their extent (centered at the query point) intersects with other hyperplanes. In this case, potential NNs may exist on those sides of the tree where they need to be searched.[40]

A k-d tree is a data structure based on hierarchical spatial decompositions. Each nonterminal node is associated with a d-dimensional cell and the subset of records within this cell. The fundamental design issue is the choice of the splitting hyperplane. The standard split method uses the data distribution by determining hyperplanes orthogonal to the median coordinate along which the points have the most significant spread. However, it generates elongated cells in the case of clustered data, resulting in longer searching times. On the other hand, the midpoint split method uses the cell's shape by dividing it through its midpoint using a hyperplane orthogonal to the longest side, creating tangibly less elongated cells. Nevertheless, many of the resulted cells can be empty, affecting tree sizes and processing times, especially when dealing with large high-dimensional data. The sliding midpoint method partitions data as the midpoint split method does. However, when it produces empty cells with no data records, it shifts the splitting plane toward data location until the plane touches the first record. This performance overcomes generating sequences of skinny or repetitive blank cells as in the standard and midpoint split methods, respectively.[43]

We used the "cKDTree" function[42] embedded in the "scipy.spatial" library in this work. The function constructs a k-d tree for quick NN lookup. We kept its default values: the sliding midpoint partitioning technique and bucket size of 16 records. Meaning that a node turns to a terminal node (leaf) if 16 points or less are associated with it. Otherwise, the algorithm continues partitioning the data.[43]

The color-based segmentation algorithm calculates the radiometric distance (RD) between two LiDAR points (p_1 and p_2) using the Euclidean norm; the square root of the sum of the squares of the differences between the R, G, and B values of each point. We added the NIR figures as follows:

$$\mathrm{RD}(p_1, p_2) = \sqrt{((R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2 + (\mathrm{NIR}_1 - \mathrm{NIR}_2)^2)}. \quad (4)$$

Equation (4) was also applied to calculate the RD between two segments (s_1 and s_2). In this case, the spectral figures were the average of each segment's points' R, G, B, and NIR values. We normalized the four features' values to have the same range (0 to 255), knowing that the radiometric resolution of the aerial image is 8 bit. We normalized the spectral characteristics before applying Eq. (4), following the rescaling (min–max normalization) equation below:

$$x_{\mathrm{normalized}} = \frac{(x - d_{\min})(r_{\max} - r_{\min})}{(d_{\max} - d_{\min})} + r_{\min}, \quad (5)$$

where x is the figure to be normalized, $d_{\min}$ is the minimum feature value, $d_{\max}$ is the maximum feature value, $r_{\min}$ is the minimum range value (0), and $r_{\max}$ is the maximum range value (255).

We kept the below threshold values as applied in Ref. 18:

- Distance point threshold (distPtThrd) = 30 cm: maximum 3D distance between *ST*'s *pt* and its neighbors.
- Distance segment threshold (distSegThrd) = 50 cm: maximum 3D distance between two neighboring segments.
- Radiometric segment threshold (radSegThrd) = 10: maximum RD between two segments.
- Minimum points per segment (segMinPoints) = 10 points.

However, we made the following modifications to the algorithm to fit the research objectives:

1. We decreased the radiometric point threshold (radPtThrd), which represents the maximum RD between two LiDAR points, from 35 to 30. The reason was to achieve a more robust object segmentation, especially between through streets and boulevards that were visually noticed to share close spectral characteristics after data registration.
2. To search for a point's neighbors using the cKDTree function, we queried the *k-d* tree for all point's NNs within the distance threshold ranges (i.e., distPtThrd and distSegThrd). Zhan et al.[18] adopted the other way around by limiting the number of looked-up NNs then checking their distances from the query point, as it provides a substantial gain in the cKDTree's efficiency.[42] However, the data size we processed in this study is significantly larger than Zhan et al.'s,[18] fundamentally slowing down the processing time. Hence, we tried to speed it up by examining all potential neighbors for radiometric similarity at once. In this way, the possibility of assigning more points to a segment and merging more segments to a segment increases. We were encouraged to develop this adjustment by the searching ranges (distPtThrd and distSegThrd), which are already confined concerning the data size and the study region's coverage area. In addition, our change checks against void lists of NNs and proceeds accordingly, as Zhan et al.[18] defend this detail. Therefore, our alternation should not affect the approach's robustness. We did not apply this adjustment in the refinement process since segments less than minimum points per segment (segMinPoints) look for merging with their NN segment without a distance constraint.
3. In the growing step, we checked the colorimetric similarity between the stack's top points' NNs and the point that initiated the segment instead of the top points themselves. Our motive was to maintain a maximum spectral difference equals to the radPtThrd value among the entire points within the same segment for a more robust point segmentation.
4. We added a condition to the growing step, keeping maximum points per segment (segMa – xPoints) below 100 points. This size control was mandatory to prevent radiometrically similar elements from being targeted in the same segments, leading to subsequent misclassification problems. To illustrate, the forestry area in the northeast direction of the study region contains trees that barely vary in color since they share

almost the same land cover, but they do vary in height. Hence, when a LiDAR point in this area initiates a segment, the segment expands gradually as per the distPtThrd value to entirely include the whole greeny area. Consequently, discriminating subclasses in the forestry area (i.e., trees, shrubs, bushes, and grass) would not be possible. This procedure of uncontrolled segment growing contradicts the research objectives, as the study proposes LiDAR-imagery data integration fundamentally to segregate subclasses that a single data type cannot separate. Hence, we introduced this adjustment as a restriction to the segment growing step.

3.3 *Feature Space Construction*

A 10D point features vector represents each point in the LiDAR dataset as below:

$$\mathrm{PF} = [\, x \quad y \quad h \quad I_1 \quad I_2 \quad I_3 \quad R \quad G \quad B \quad \mathrm{NIR}\,]^T, \tag{6}$$

where PF is the point features, x and y are the point's x and y coordinates, respectively, h is the point height calculated in the ground filtering process, I_1, I_2, I_3 are the point's intensities obtained from the multispectral LiDAR sensor in the C_1, C_2, and C_3 channels, respectively, and R, G, B, NIR are the point's spectral properties inherited after LiDAR-aerial data registration in the red, green, blue, and near-infrared bands, respectively.

These point features fundamentally build the following radiometric and geometric features per LiDAR segment.

3.3.1 *Radiometric features*

The following is a 25D radiometric segment features vector that we calculated for each 3D LiDAR segment:

$$\begin{gathered}\mathrm{SF_{rad}} = [\mu I_1\ \mu I_2\ \mu I_3\ \mu R\ \mu G\ \mu B\ \mu\mathrm{NIR}\ Br \\ \mathrm{ratio_{I_1}}\ \mathrm{ratio_{I_2}}\ \mathrm{ratio_{I_3}}\ \mathrm{ratio_R}\ \mathrm{ratio_G}\ \mathrm{ratio_B} \\ \mathrm{ratio_{NIR}}\ \mathrm{NDVI}\ \mathrm{EVI}\ \mathrm{GLI}\ \mathrm{GNDVI}\ \mathrm{GARI} \\ \mathrm{MSAVI_2}\ \mathrm{MNLI}\ \mathrm{TDVI}\ \mathrm{VrNIR_{BI}}\ \mathrm{VgNIR_{BI}}]^{\mathrm{T}}\end{gathered}, \tag{7}$$

where $\mathrm{SF_{rad}}$ is the radiometric segment features vector, and $\mu I_1, \mu I_2, \mu I_3, \mu R, \mu G, \mu B, \mu$NIR are the segment's mean $\mathrm{I_1}$, $\mathrm{I_2}$, $\mathrm{I_3}$, R, G, B, and NIR values, respectively, calculated by

$$\begin{aligned}\mu I_1 &= \frac{1}{N}\sum_{i=1}^{N} I_{1_i}, \\ \mu I_2 &= \frac{1}{N}\sum_{i=1}^{N} I_{2_i}, \\ \mu I_3 &= \frac{1}{N}\sum_{i=1}^{N} I_{3_i} \\ \mu R &= \frac{1}{N}\sum_{i=1}^{N} R_i, \\ \mu G &= \frac{1}{N}\sum_{i=1}^{N} G_i, \\ \mu B &= \frac{1}{N}\sum_{i=1}^{N} B_i, \\ \mu\mathrm{NIR} &= \frac{1}{N}\sum_{i=1}^{N} \mathrm{NIR}_i,\end{aligned} \tag{8}$$

where N is the number of points the segment includes, i is a counter that runs over N, and Br is the brightness value that averages the segment's seven colors, given as

$$Br = \frac{1}{7}(\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}), \tag{9}$$

$\mathrm{ratio}_{\mathrm{I}_1}, \mathrm{ratio}_{\mathrm{I}_2}, \mathrm{ratio}_{\mathrm{I}_3}, \mathrm{ratio}_{\mathrm{R}}, \mathrm{ratio}_{\mathrm{G}}, \mathrm{ratio}_{\mathrm{B}}, \mathrm{ratio}_{\mathrm{NIR}}$ are the segment's ratios of I_1, I_2, I_3, R, G, B, and NIR, respectively, given as

$$\begin{aligned}
\mathrm{ratio}_{I_1} &= \frac{\mu I_1}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}} \\
\mathrm{ratio}_{I_2} &= \frac{\mu I_2}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}} \\
\mathrm{ratio}_{I_3} &= \frac{\mu I_3}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}} \\
\mathrm{ratio}_{\mathrm{R}} &= \frac{\mu R}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}} \\
\mathrm{ratio}_{\mathrm{G}} &= \frac{\mu G}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}} \\
\mathrm{ratio}_{\mathrm{B}} &= \frac{\mu B}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}} \\
\mathrm{ratio}_{\mathrm{NIR}} &= \frac{\mu \mathrm{NiR}}{\mu I_1 + \mu I_2 + \mu I_3 + \mu R + \mu G + \mu B + \mu \mathrm{NIR}}.
\end{aligned} \tag{10}$$

The mean colors, brightness, and color ratio features represented in Eqs. (8)–(10) are inspired by Kang et al.'s study.[12] However, we included additional vegetation indices for better segregation of trees and grasses, which the scene contains. The added indices are part of multiple broadband greenness vegetation indices offered by ENVI software[44] and chosen based on their compatibility with the nature of the study area. They are computed as follows:

$$\mathrm{NDVI} = \frac{\mu \mathrm{NIR} - \mu R}{\mu \mathrm{NIR} + \mu R}, \tag{11}$$

where NDVI is the segment's normalized difference vegetation index that identifies healthy and green vegetation

$$\mathrm{EVI} = \frac{2.5(\mu \mathrm{NIR} - \mu R)}{\mu \mathrm{NIR} + 6\mu R - 7.5\mu B + 1}, \tag{12}$$

where EVI is the segment's enhanced vegetation index. It improves NDVI in areas of high vegetation by accommodating soil background signals and atmospheric effects:

$$\mathrm{GLI} = \frac{2\mu G - \mu R - \mu B}{2\mu G - \mu R + \mu B}, \tag{13}$$

where GLI is the segment's green leaf index, designed initially for digital RGB images

$$\mathrm{GNDVI} = \frac{\mu \mathrm{NIR} - \mu G}{\mu \mathrm{NIR} + \mu G}, \tag{14}$$

where GNDVI is the segment's green normalized difference vegetation index. It is more sensitive to chlorophyll concentration than NDVI, as it uses the green instead of the red spectrum:

$$\mathrm{GARI} = \frac{\mu \mathrm{NIR} - \mu G + \gamma(\mu B - \mu R)}{\mu \mathrm{NIR} + \mu G - \gamma(\mu B - \mu R)}, \tag{15}$$

where GARI is the segment's green atmospherically resistant index. It is more susceptible to a wide range of chlorophyll concentrations and less sensitive to atmospheric impacts than NDVI, as GARI involves a weighting function (γ constant = 1.7) that depends on aerosol conditions in the atmosphere

$$\mathrm{MSAVI}_2 = \mu\mathrm{NIR} + 0.5 - 0.5(\sqrt{(2\mu\mathrm{NIR}+1)^2 - 8(\mu\mathrm{NIR} - \mu R)}), \tag{16}$$

where MSAVI_2 is the segment's second modified soil adjusted vegetation index that decreases soil noise to highlight healthy vegetation:

$$\mathrm{MNLI} = \frac{((\mu\mathrm{NIR})^2 - \mu R))(1+L)}{(\mu\mathrm{NIR})^2 + \mu R + L}, \tag{17}$$

where MNLI is the segment's modified nonlinear index that accounts for the soil background by including L; a canopy background adjustment factor of value 0.5, and

$$\mathrm{TDVI} = \frac{1.5(\mu\mathrm{NIR} - \mu R)}{\sqrt{((\mu\mathrm{NIR})^2 + \mu R + 0.5)}}, \tag{18}$$

where TDVI is the segment's transformed difference vegetation index that detects green covers in urban morphologies.

Moreover, we computed two urban indices that are designed for extracting built-up areas as below:[45]

$$\mathrm{VrNIR}_{\mathrm{BI}} = \frac{\mu R - \mu\mathrm{NIR}}{\mu R + \mu\mathrm{NIR}} \quad \mathrm{VgNIR}_{\mathrm{BI}} = \frac{\mu G - \mu\mathrm{NIR}}{\mu G + \mu\mathrm{NIR}}, \tag{19}$$

where $\mathrm{VrNIR}_{\mathrm{BI}}$ and $\mathrm{VgNIR}_{\mathrm{BI}}$ are the visible red-based and green-based built-up indices, respectively.

The mean values of the I_1, I_2, I_3, R, G, B, and NIR that appear in Eqs. (9)–(19) were calculated after the normalization of the corresponding point features in Eq. (6) to range from 0 to 255 by substituting in Eq. (5).

3.3.2 *Geometric features*

Below is a 12D geometric segment features vector that we calculated for each 3D LiDAR segment. They are a combination of what Kang et al.[12] and Martin et al.[46] have applied in their studies:

$$\mathrm{SF}_{\mathrm{geom}} = [\mu h\, h_{\mathrm{var}}\, \mathrm{plane}_{\mathrm{res}}\, \lambda_1\, \lambda_2\, \lambda_3\, A_\lambda\, P_\lambda\, S_\lambda\, L_\lambda\, O_\lambda E_\lambda]^T, \tag{20}$$

where $\mathrm{SF}_{\mathrm{geom}}$ is the geometric segment features vector, and $\mu h, h_{\mathrm{var}}$ are the segment's mean height and height variance, respectively, given as

$$\mu h = \frac{1}{N}\sum_{i=1}^{N} h_{\mathrm{i}}, \tag{21}$$

$$h_{\mathrm{var}} = \frac{1}{N}\sum_{i=1}^{N} (h_i - \mu h)^2, \tag{22}$$

$\mathrm{plane}_{\mathrm{res}}$ is the plane residual represented by the Euclidean distance norm of the vector that contains the segment's points' residuals from their best-fitting plane estimated by least-squares, $\lambda_1, \lambda_2, \lambda_3$ are the eigenvalues resulting from the eigendecomposition of the segment's covariance matrix, which is constructed from the covariances between each pair of the segment's x, y, and h point features [Eq. (6)]. For instance, the covariance between x and y is computed as

$$\sigma_{xy} = \frac{\sum_{i=1}^{N}(x_{\mathrm{i}} - \mu x)(y_{\mathrm{i}} - \mu y)}{N-1}, \tag{23}$$

where $\mu x, \mu y$ are the segment's mean x and y coordinates, respectively.

The segment's three eigenvalues are normalized to accommodate different scales as follows:

$$\lambda_j = \frac{\lambda_j}{\sum_{j=1}^{3} \lambda_j}. \tag{24}$$

They are sorted in a descending order so $\lambda_1 > \lambda_2 > \lambda_3$, and the following eigenvalue-based geometric features are calculated:

$$\begin{aligned} A_\lambda &= \frac{\lambda_1 - \lambda_3}{\lambda_1} \\ P_\lambda &= \frac{\lambda_2 - \lambda_3}{\lambda_1} \\ S_\lambda &= \frac{\lambda_3}{\lambda_1} \\ L_\lambda &= \frac{\lambda_1 - \lambda_2}{\lambda_1} \\ O_\lambda &= \sqrt[3]{\lambda_1 \lambda_2 \lambda_3} \\ E_\lambda &= -\sum_{j=1}^{3} \ln \lambda_j, \end{aligned} \tag{25}$$

where A_λ is the segment's anisotropy that exposes how its properties are directional (differ with different directions), in opposite to isotropy, where properties are uniformly distributed in all directions, P_λ, S_λ, L_λ are measures of the segment's planarity, sphericity, and linearity, respectively, O_λ is the segment's omnivariance that describes its 3D shape, and E_λ is the segment's eigenentropy.

3.4 *Feature Space Dimensionality Reduction Using PCA*

We used the "PCA" class in the "decomposition" module embedded in the "scikit-learn" library in Python to implement the PCA.[47] The class implicitly applies the five steps previously mentioned in Sec. 2.4. We created a PCA model and set the number of components to the exact dimensions of the original data. Then, we fitted the model to the input feature space. Afterward, we targeted the components with the highest accumulating variances as the most significant PCs, by which we recreated and refitted the model and finally identified the subspace dimensions. Lastly, we transformed (projected) the original data to the determined output space.

There is no way to name the most significant features within a dimensionality reduction technique; however, the PCA enables recognizing the most meaningful features contributing to each PC's creation. The "components" attribute in a PCA model is a matrix whose rows resemble the PCs, and columns mirror the features in the input space. Each value in the components matrix represents the weight a feature participated with while producing a PC. By sorting each row in descending order, we could reflect on the most critical features that constructed each PC. We carried out the dimensionality reduction for the ground and nonground subsets separately.

3.5 *LiDAR Data Classification Using MLP Neural Networks*

We collected data for classification model training, validation, and testing manually on ERDAS IMAGINE by digitizing polygons of points for each observed class. We acquired 13,953 segments, 80% of which contributed to training and validating the MLP neural network classification model, with a ratio of 80% to 20%, respectively. We used the remaining 20% to test the model (test 1). To ensure a robust model consistency, we added second testing (test 2) using a set of 5337 points collected in the same way. Figure 7 shows the percentage breakdown of the obtained data. It is worth mentioning that the validation and both testing data are three different sets and did not participate in training the model. Also, we maintained a class representation in

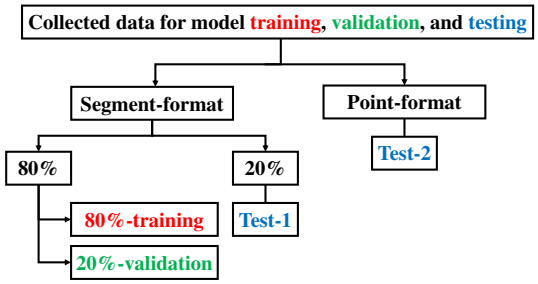


Fig. 7 Percentage breakdown of collected data: training, validation, and testing.

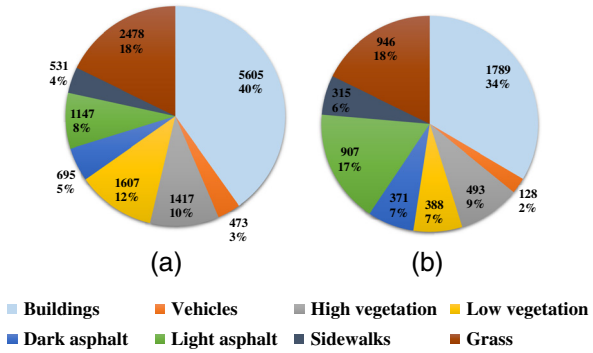


Fig. 8 Percentage/numerical breakdown of collected data: eight classes. (a) Collected segments (splitted for training, validation, and test 1). (b) Collected points (used in test 2).

these four splits proportional to its size in the collected data. Equivalent class representation ensures good model learning and unbiased evaluation. Figure 8 shows the categorization of the eight classes within the collected data.

Since the class feature in the training, validation, and testing samples belongs to categorical data as a multiclass prediction problem, we converted the class feature to numerical values in two steps. First, we applied label encoding to assign an integer figure to each category (class). Then, we implemented one-hot encoding to give those figures a binary representation to eliminate an ordinal association among the classes.[48]

We used the "Keras" Application Programming Interface (API) in Python[49] to create an MLP neural network model of four layers. The input layer's dimension was set to the number of features (37 in total if we entirely consider the radiometric and geometric features). At the same time, the two hidden layers and the output layer had 16, 12, and 4 neurons, respectively. We ran the classification of ground and nonground segments independently; hence, the four neurons in the output layer represent the number of classes observed in both sets. The model was compiled using the Adam optimizer[50] and the categorical cross-entropy loss function.[51] We applied a traditional feedforward network, where forward processing carried out an upward activation of the neurons until the final output. The rectified linear unit function[52] activated the input and hidden layers, whereas the softmax function activated the output layer.[52] Softmax is typically used in multiclass prediction applications, as it provides the probability membership of each segment to belong to each of the output classes. We assigned the class of the highest probability to each segment. The loss function computed the error and backpropagated it, while the optimizer updated the weights according to their contribution to the error. The error backpropagation was iterated over 100 epochs and 50 batches per epoch until the model arrived at a set of weights minimizing the prediction error.[30]

We examined the following 10 scenarios, each of which trains an MLP neural network model on a bundle of the 37 features, as follows:

1. μ h, μ I_1.
2. μ h, μ I_2.

3. μ h, μ I_3.
4. μ h, μ I_1, μ I_2.
5. μ h, μ I_1, μ I_3.
6. μ h, μ I_2, μ I_3.
7. μ h, μ I_1, μ I_2, μ I_3.
8. μ h, μ I_1, μ I_2, μ I_3, μ R, μ G, μ B, μ NIR.
9. SF_{rad}, SF_{geom}.
10. PCs.

Scenarios (1) to (7) thoroughly study the capabilities of the multispectral LiDAR features in urban classification when combined with the height values in seven different ways. Scenario 8 reveals the effect of including R, G, B, and NIR from the aerial image on classifying the scene, whereas scenario 9 tests the hypothesis of combining additional calculated radiometric and geometric features on enhancing the classification results. Finally, scenario 10 attempts a lower dimensionality represented by the most significant PCs instead of the whole input space in the preceding scenario. To ensure a consistent assessment of the 10 scenarios, the training, validation, and testing datasets were the same for the 10 MLP classification models.

3.6 *Classification Assessment*

We validated the 10 classification models using different classification metrics and resampling techniques. We constructed the accuracy matrix in test 1 and test 2 for each MLP model to calculate the accuracy, precision, recall, and F_1-score metrics. They are explained below for a binary classification that deals with two classes: positive and negative (Fig. 9). We accommodated them accordingly in the calculations to fit a multiclass prediction problem:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}, \tag{26}$$

where Accuracy is the model's overall accuracy that donates the fraction of the total samples that are correctly classified, TP is the true positive, reflecting the number of positive samples that the model correctly predicts as a positive class,

TN is the true negative, reflecting the number of negative samples that the model correctly predicts as a negative class, FN is the false negative, reflecting the number of positive samples that the model incorrectly predicts as a negative class, and FP is the false positive, reflecting the number of negative samples that the model incorrectly predicts as a positive class,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{27}$$

where Precision is the precision of the positive class, referring to the fraction of positive predictions that are positive in reality. It is calculated for each class

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{28}$$

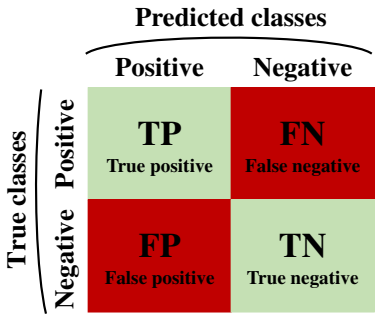


Fig. 9 Confusion matrix of binary classification of positive and negative classes.

where Recall is the recall of the positive class, referring to the fraction of positive samples that are correctly predicted positive. It is calculated for each class, and

$$F_1\text{-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{29}$$

where F_1-score is the harmonic mean of the precision and recall of the positive class. It is calculated for each class.

We trained, validated, and tested the 10 classification models using the same training, validation, and testing sets, respectively, which are illustrated in Sec. 3.5 to maintain an unbiased comparison. In addition, we applied the k-fold cross-validation as a different resampling technique. It divides the training records into a k number of folds, where one fold participates as a testing set and the rest contribute to training the model. The algorithm runs k times, and at each turn, a different fold takes part as the held-out testing set. In this way, the entire samples contribute to the fitting and evaluation processes, ensuring robust assessment figures. The k-fold cross-validation technique results in a k number of MLP models, whose accuracies are averaged and their standard deviations are calculated.[30,53] In this study, we set k to 10, used the segmented training dataset (Fig. 7), constructed stratified folds to guarantee all classes in training and validation folds are represented proportionally to their size in the training dataset, and repeated the algorithm 100 times.

4 Results and Discussions

4.1 *Dimensionality Compression of Feature Space by PCA*

The segment growing step in the color-based segmentation process resulted in 38,930 and 92,489 rough segments for the ground and nonground LiDAR points. The merging and refinement step reduced the number of segments to 21,853 and 36,138, respectively, which is 57,991 total segments for the entire point cloud.

After computing the 37D feature space for each segment, we commenced the PCA with 37 PCs, the size of the input space. By sorting the variances in descending order, we calculated the accumulated variance as a preparatory step toward the definite number of components to consider. Figure 10 shows that close to 100% cumulative variance is achieved by recognizing only nine and seven PCs in the analysis of nonground and ground data, respectively. Hence, we reperformed the PCA considering these most significant components and projected the 37D input feature space of the nonground and ground LiDAR points into a 9D and 7D output space, respectively.

Figure 11 reveals the contribution of each feature in the 37D input space to creating the most significant PCs after normalizing the features' weights to range from 0 to 1, indicating no and full contribution, respectively. The most significant PCs of both ground and nonground data primarily consist of combinations of features of direct LiDAR and imagery spectral intensities; I_1, I_2, I_3, R, G, and NIR, with a notable contribution from EVI. As height is an expected feature to distinguish between nonground objects, it is not surprising that it also appears to dominate the most significant PCs of the nonground points. This initial overview highlights anticipation of more influential spectral features than the derived geometric characteristics when the data are classified.

4.2 *Classification of LiDAR Data Using MLP Neural Networks*

An MLP neural network is a stochastic machine learning algorithm whose decisions vary randomly during the learning process. It uses random initial weights and random shuffle of samples (batches) at each epoch to help the model seek better solutions by avoiding local or deceptive optima. Consequently, each time an MLP neural network algorithm runs, it creates a different model, provides different predictions, and produces a different accuracy. These variations occur even when the algorithm uses the same training data each time it runs.[54] Hence, it is essential to test the MLP models on different datasets to ensure compatible results.

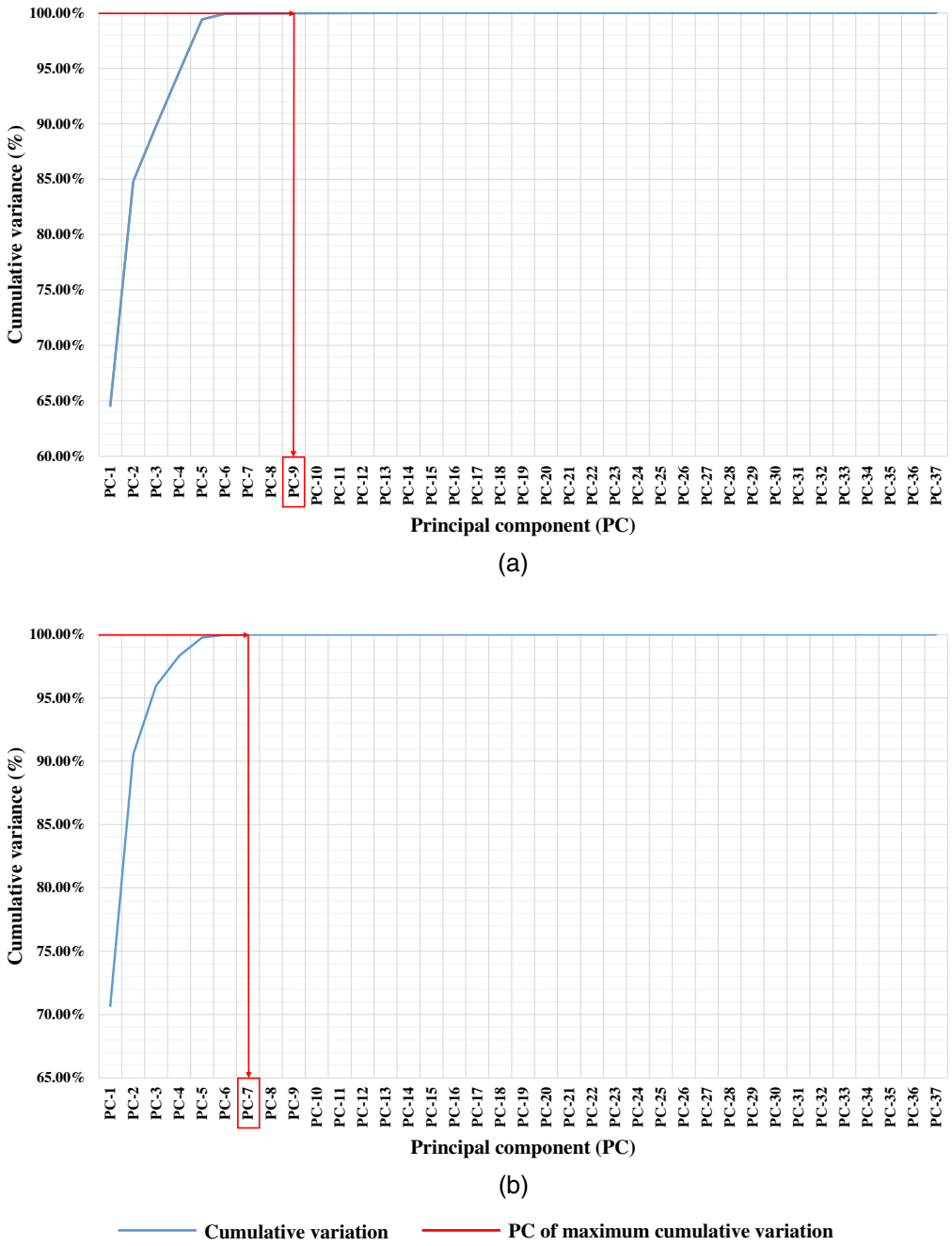


Fig. 10 PCs' cumulative variances. (a) Nonground LiDAR data. (b) Ground LiDAR data.

Figure 12 shows the overall mapping accuracies of the 10 scenarios using the validation, test 1 and test 2 splits (previously addressed in Sec. 3.5), in addition to the *k*-fold approach. In the *k*-fold approach, different nonoverlapping subsets of the training data, which summed up to the entirety of the training set, participated in the validation in rotation. The fourth bar in each scenario represents the mean accuracy of the 10 folds in the 100 repetitions when each fold acted as the held-out testing set. The values above describe the standard deviation of the 1000 accuracy values.

In each scenario, the four evaluations are close to each other, indicating consistent MLP models that are reliable to apply on unseen data for predictions. However, the first and third models show relatively lower accuracy figures when verified by the test 2 set. This decrease is probably the result of insufficient input feature spaces in both scenarios that allowed misclassified segments, of which some points of test 2 are accidentally part. Test 2 is a set of points,

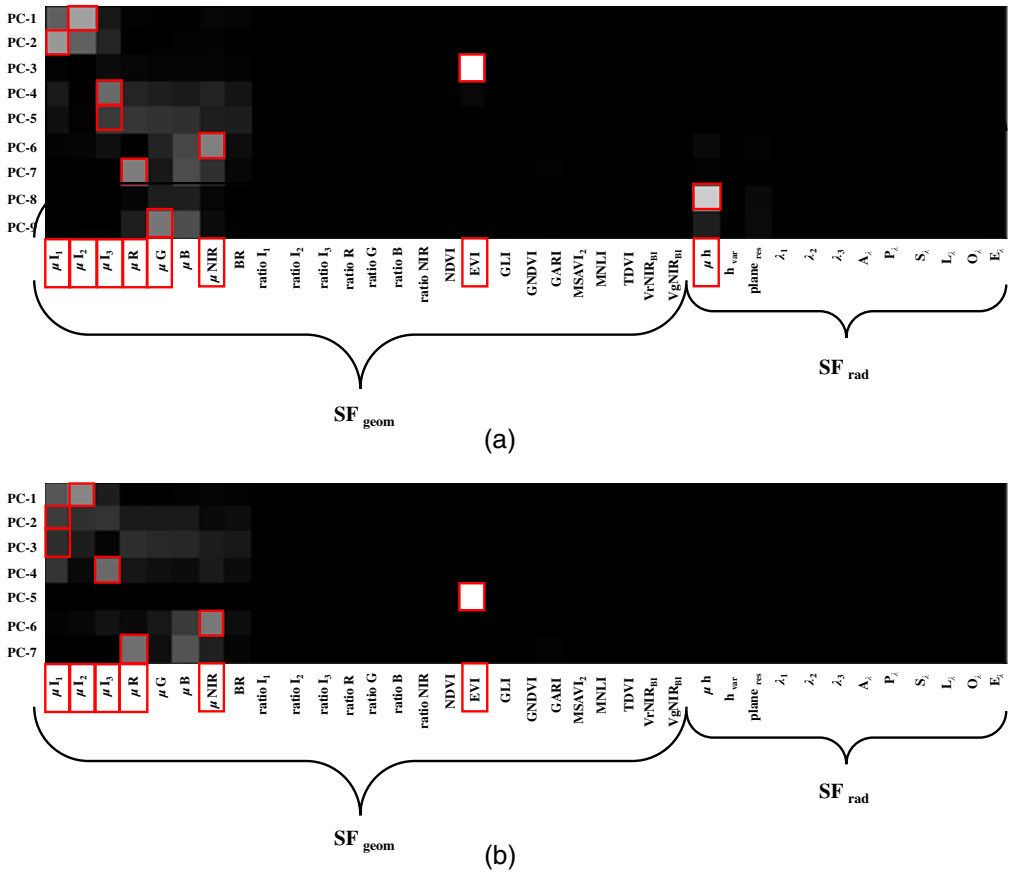


Fig. 11 The most significant PCs' most contributing features. (a) Nonground LiDAR data. (b) Ground LiDAR data.

not segments as the rest of the assessment splits are, so the misclassification effect is more pronounced. The evaluation on test 2 shows consistency with the other three assessment splits in the remaining scenarios whose feature spaces are larger with lower standard deviations, which supports this argument. The remaining part of this section discusses the classification results based on the models' evaluation using the test 1 dataset.

Figure 13 shows the classification results using the test 1 data split. A general glance at the overall mapping accuracy shows its gradual increase by including more features in the input space of the classification. However, there is a significant leap in the nonground accuracy compared with the ground one highlighted in the first three scenarios. This notable increase results from the

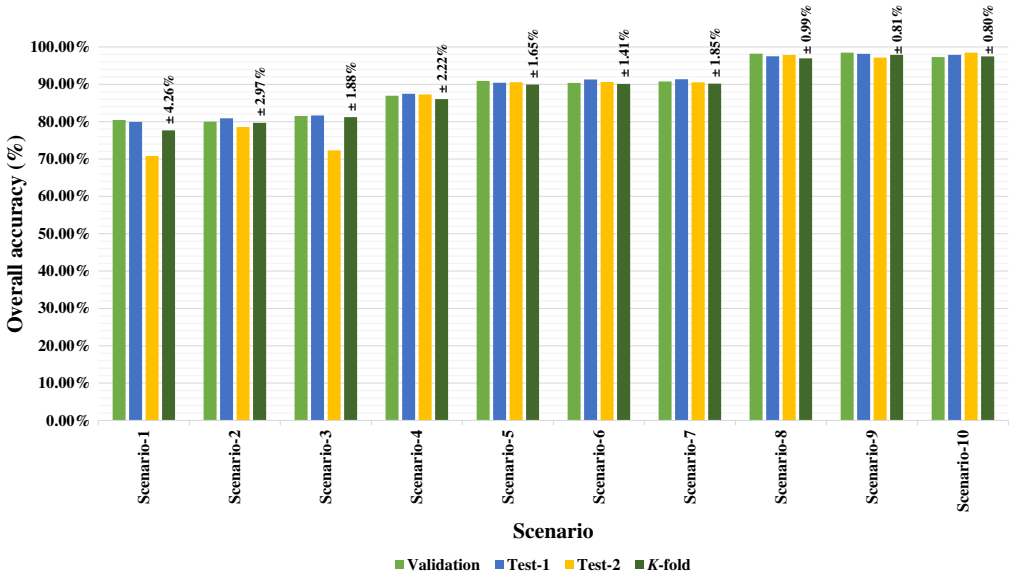


Fig. 12 Assessment of LiDAR data's multilevel object-based classification: validation, testing, and *k*-fold results.

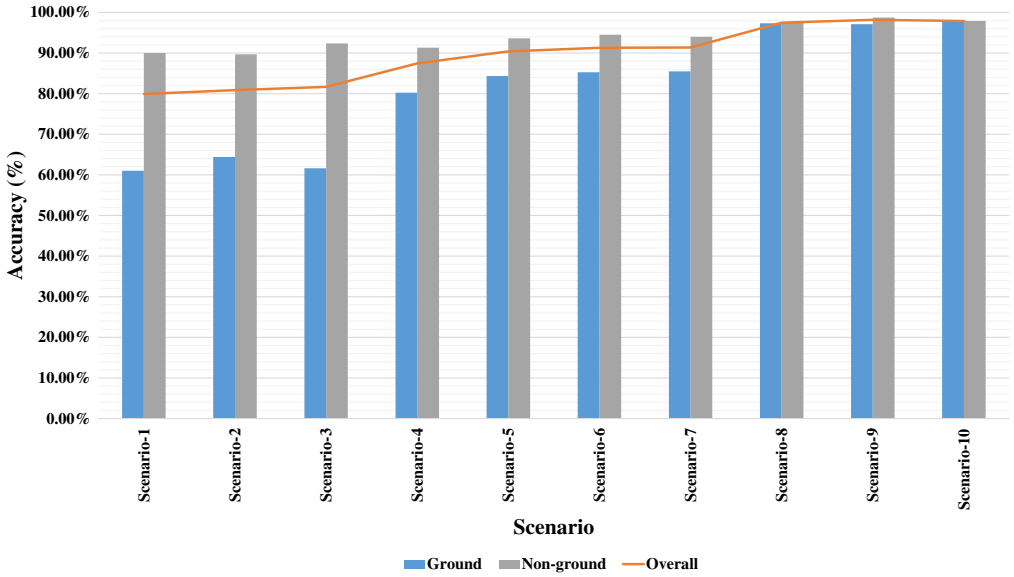


Fig. 13 Accuracy of LiDAR data's multilevel object-based classification (test 1 set): ground, non-ground, and overall accuracies.

height being a discriminative feature in the classification of nonground data, whose accuracy is around 90% when combined with a single LiDAR channel. On the contrary, the height range of the ground data is 4 cm, which does not provide room for a better classification when combined with a LiDAR spectral channel, leading to a ground classification accuracy of around 60%.

When comparing the first three scenarios, the nonground accuracies of scenario 1 and scenario 2 are almost the same ($\approx$ 90%); however, scenario 3's is relatively larger (92.37%) because of its higher capability in detecting vehicles. On the other hand, the ground accuracies of scenario 1 and scenario 3 are close ($\approx$ 61%); however, scenario 2's ground accuracy is relatively larger (64.40%) because of its higher capability in detecting light asphalt. Nevertheless, these variances slightly affect the three scenarios in total; i.e., 79.92%, 80.89%, and 81.68%, respectively.

The following three scenarios reveal the impact of alternating two of the three LiDAR channels. Adding a second LiDAR spectrum booms the overall accuracy to 87.47%, 90.41%, and

91.30% in scenario 4, scenario 5, and scenario 6, respectively, as it relatively enhances the predictions of the vehicles, dark, and light asphalts. The increase of the overall accuracy is vitally contributed by the ground classification, which rushes to around 80% compared with around 60% in the past three scenarios. Combining a second LiDAR beam compensates partially for the idle height feature in the ground classification as per the first three scenarios, with steady progress in the nonground classification figures. Combining the three LiDAR channels in scenario 7 does not add to the highest accuracies of a dual-channel inclusion provided by scenario 6.

Scenario 8 renders another remarkable development in the classification results. Introducing the aerial photo's radiometric properties (R, G, B, and NIR) increases the accuracy to above 97%. The added spectral features push the nonground and ground accuracies to 97.59% and 97.33%, respectively, striking the overall accuracy of the scenario to 97.49%. This increase results from a continuous improvement in the vehicle, dark, and light asphalt classes.

By accounting for the entire 37D feature space in scenario 9, the nonground classification keeps growing to reach an accuracy of 98.74%. The height-derived geometric feature space allows for better vehicles predictions, raising the accuracy of the nonground classification. However, the full-feature input space slightly affects the dark and light asphalt classes, lightly decreasing the ground accuracy to 97.12%, making the scenario's overall accuracy 98.17%, and the highest among the entire 10 scenarios.

The most significant PCs in scenario 10 insignificantly lower the nonground classification to 97.91% due to a decrease in the vehicles class accuracy, which is intuitively expected when only a subset of the components participates in the classification. However, the selected components yet contain the most distinguishing characteristics. Unlike the nonground classification, scenario 10 slightly enhances the ground classification to 97.84% due to an increase in the dark and light asphalt classifications. Despite the nontangible improvement, it suggests that the original input space may include a feature or more with a negative impact on the classification model that is eliminated in the PC space, enhancing the results in consequence. Scenario-10 ends with an overall accuracy of 97.89%.

Figure 14 digs more into the classification results by visualizing the per-class accuracies of the different scenarios represented by the F_1-score. If a class's F_1-score is zero, it means either its precision or recall is zero. Height and C_1 LiDAR channel in scenario 1 can reasonably differentiate buildings, high vegetation, and low vegetation with 95.16%, 80.95%, and 87.61%, respectively, justifying the scenario's high nonground accuracy (Fig. 13). However, the two features show buildings/high-vegetation and vehicles/low-vegetation misclassification problems.

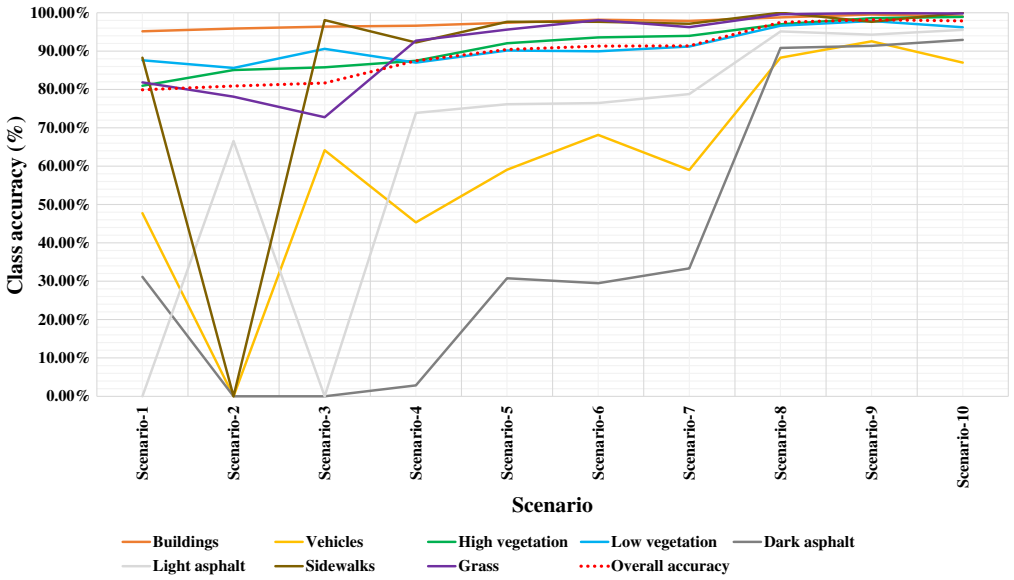


Fig. 14 Per-class accuracies of LiDAR data's multilevel object-based classification: F_1-score (test 1 set).

Because of the different data acquisition times (LiDAR and imagery data were obtained independently in 2015 and 2014, respectively), the model misclassifies many high vegetation segments as buildings. Some high-vegetation LiDAR points resemble asphalt locations on the aerial image, where trees were not planted. The model also misclassifies some segments of buildings as high vegetation due to orthorectification problems. These problems are attributed to the building facades appearing in the aerial image; consequently, LiDAR segments on roof edges inherited the spectra of building interfaces and surrounding surfaces usually planted in residential areas.

More severely, scenario-1 shows poor vehicle classification results (47.76%) as it mixes a substantial part of the class with low vegetation. It is unlikely for a vehicle to keep the exact location in a study scene, particularly when captured by two different sensors on different dates. Therefore, they inherited the corresponding spectra of the ground surfaces where they usually park (i.e., grassy sidewalks). Hence, the abovementioned pairs of misclassified classes are radiometrically and geometrically indistinguishable, with the height information being the solely geometric feature in scenario 1.

On the other hand, scenario 1 recognizes sidewalks and grass with an accuracy of 88.29% and 81.85%. Nevertheless, it reports dark-asphalt/light-asphalt/grass misclassification that plunges dark asphalt to 31.14% and completely misses light asphalt. Scenario 2 introduces C_2 LiDAR channel instead of C_1. It increases the high-vegetation accuracy (85.08%), almost does not change the accuracy of buildings (95.89%), decreases the accuracy of low vegetation (85.60%), and completely drops the accuracy of vehicles due to the previously mentioned misclassifications, which eventually does not alter the nonground classification as noticed in Fig. 13. Nonetheless, scenario 2 identifies light asphalt with an accuracy of 66.56% after a drop in scenario 1. This jump enhances the ground accuracy (Fig. 13) despite the misclassification of the entire four ground classes, which misses the sidewalks and dark asphalt, and lowers the grass accuracy to 78.10%.

Scenario 3 tests C_3 instead of the other two LiDAR channels: C_1 and C_2. The model boosts the vehicle classification to 64.15% after a drop in scenario 2, consequently raising the low vegetation to 90.59%. By providing a slight increase to the classification of buildings (96.40%), scenario 3 increases the nonground classification accuracy compared with scenario 1's and scenario 2's (Fig. 13). It also records a hit in classifying sidewalks (98.10%), compensating for missing both asphalts as grass, lowering the grass accuracy to 72.78%, in consequence. This hit keeps scenario 3's ground accuracy close to scenario 1's.

Alternating the three LiDAR channels in pairs as per scenario 4 to scenario 6 presents a nearly steady increase of all classes, converging in a high accuracy range, from 87% to 98%, except for vehicles and asphalts. The inclusion of dual channels rushes the accuracy of light asphalt to 73.89%, 76.16%, and 76.47% in the three scenarios, respectively. The dark asphalt class accuracy also jumps from 2.84% in scenario 4 to 30.77% and 29.47% in scenario 5 and scenario 6, respectively. On the other hand, the vehicles' accuracy decreases to 45.33% in scenario 4, then raises to 59.06% and 68.16% in scenario 5 and scenario 6, respectively. These figures give scenario 6 higher ground, nonground, and overall accuracy than the preceding scenarios and are very close to scenario 7's, donated by including the three LiDAR channels (Fig. 13). Compared with scenario 6, scenario 7 decreases vehicles from 68.16% to 58.99% and increases dark asphalt from 29.47% to 33.33%.

The last three scenarios continue the nearly steady improvement of all classes, yet converging in a higher accuracy range, from 96% to full prediction, except for vehicles and both asphalts. Accumulating the aerial photo's spectra in scenario 8 tangibly improves the troubling classes to 88.30%, 90.85%, and 95.13% for vehicles, dark, and light asphalt, respectively. The R, G, and B bands lend a hand in discriminating dark and light asphalts even with the naked eye visualization [Fig. 5(e)]. At the same time, the NIR band helps identify the greeny features with 96.98%, 96.62%, and 99.60% accuracy for the high, low vegetation, and grass classes, respectively, which raises the accuracy of the vehicles and buildings (98.80%) in accordance since they are misclassified with low vegetation and high vegetation, respectively.

The SF_{geom} introduced in scenario 9 better eliminates this misclassification, as the mixed classes share close values of the SF_{rad} added in the same scenario since they already have similar green characteristics. Scenario-9 notably increases the vehicle accuracy to 92.55% and develops the accuracy of buildings, high vegetation, and low vegetation to 99.55%, 98.58%, and 97.85%,

respectively. On the other hand, the scenario suggests one or more confusing features in the SF_{rad}, such as lowering the sidewalks to 97.65% after a full prediction in scenario 8, also decreasing the light asphalt to 94.29%. The PCA eliminates confusing features by definition; therefore, scenario 10 increases the accuracy of dark, light asphalts, sidewalks, and grass to 92.94%, 95.54%, 100.00%, and 99.80%. These figures are not only higher than scenario 9 but also even exceed what scenario 8 achieves for ground classes.

4.3 *Production of Final Urban Map*

The bar charts in Fig. 15 summarize the best and the worst classification accuracy results we achieved in this study from scenario and class perspectives. Figure 15(a) displays the most useful versus the most unfavorable scenarios for the land-uses, ground, nonground, and overall classification, based on the results discussed in Sec. 4.2. Scenario 9 produces the highest nonground accuracy (98.74%) as expected after the scenario being the highest-scoring in the nonground classes, except for high vegetation. The class yields 98.94% accuracy in scenario 10, slightly above the 98.58% obtained by scenario 9. Likewise, scenario 10 provides the best ground (97.84%) and ground per-class accuracies, except for grass that reaches 99.90% in scenario 9, insignificantly over the 99.80%, it hits with scenario 10. Therefore, we considered scenario 9's and scenario 10's nonground and ground classifications to produce the final urban map.

Figure 15(b) shows the same results but from a different aspect, as it sums up the most versus the least detectable classes for each scenario. Building roofs are the best hits of the majority of the scenarios. As long as a feature space includes the height records, adding a single radiometric feature guarantees >95% detection (i.e., 95.16% in scenario 1), which can be improved by adding more features. Scenario 3 is the best to efficiently target sidewalks or similar elements (i.e., landmarking at pedestrian crossing intersections) with a >98% accuracy.

Scenario 2 and scenario 4 are still easy-to-pick options (narrow feature spaces, and thus, fast processing) if a building-accuracy higher than what scenario 1 offers is required. Scenario 4 is also beneficial in grass-oriented applications with a >92% accuracy (Fig. 14). Scenarios 5 to 7 are good choices if a >97% building detection is needed or a >90% green accuracy is attempted (>90%, >92%, >95% for low, high vegetation, or grass, respectively) [Fig. 14)].

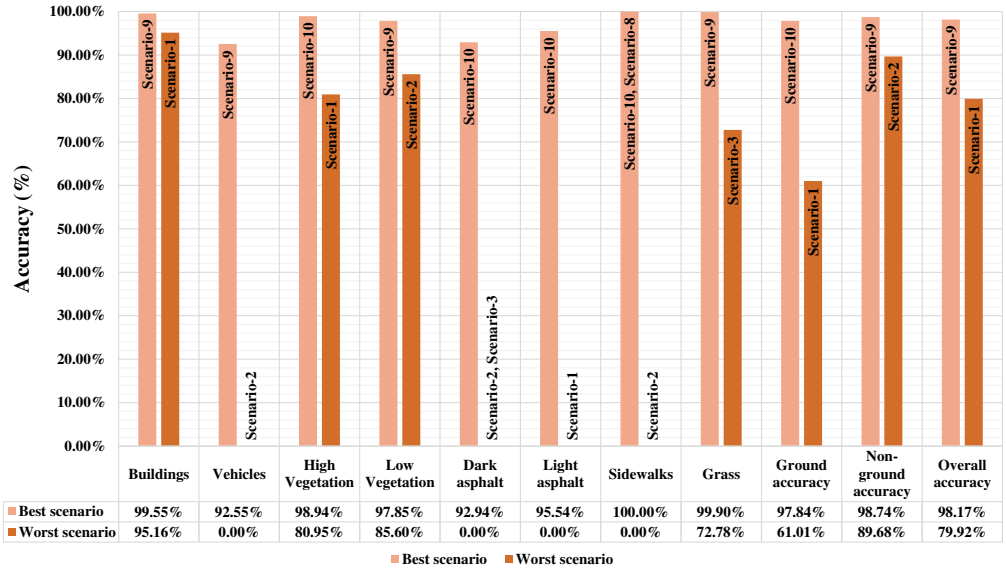
Scenario-8 is a perfect compromise of the entire eight classes. Besides a full grass detection, the inherited R, G, B, and NIR substantially solve the vehicle/low-vegetation and dark/light-asphalt misclassifications emerging in the preceding scenarios. Vehicles are the scenario's least accurate features, yet, detected with an accuracy of 88.30%. Nevertheless, scenario 9 is optimum for the maximum misclassification elimination between geometrically distinguished classes (i.e., vehicle/low-vegetation and building/high-vegetation). Consequently, the scenario fits urban mapping applications with fine accuracy requirements (>98%). Vehicles are still the scenario's least accurate features, but 91.39% accurate. Scenario 10 suits large input feature spaces when one lacks a predetermined knowledge about their significance.

We want to emphasize that the results' discussions, along with the recommended case uses of each scenario, are guidelines for researchers, assuming similar urban objects and encountered data challenges (i.e., orthorectification, shadow, and different acquisition time). Researchers should consider their data structure, observed classes, and application requirements to decide optimal classification scenarios. Our analysis may shed light on even more feature combinations and testing scenarios.

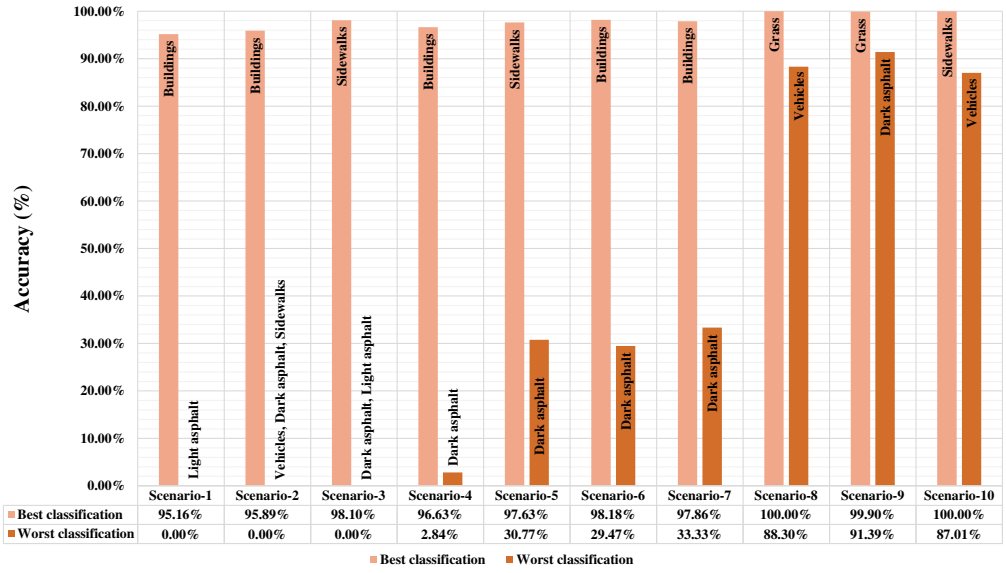
Picking the best scenario depends on the application's nature and objectives. This study aims to provide the best possible accurate urban mapping that accommodates all accuracies: per-class and overall accuracies. Consequently, we chose scenario 9 and scenario 10 for nonground and ground classifications. The combined scenarios increase the overall final map accuracy to 98.43%, slightly higher than the maximum overall accuracy achieved by scenario 9 alone [98.17%; Fig. 15(a)].

Figure 16 shows the per-class accuracy of the combined-scenario classification. It separates the nonground and ground classes accuracies of scenario 9 and scenario 10, respectively, from Fig. 14 for better visualization of the final map's per-class figures visualized in a bar chart.

Figure 17 reflects on the learning curves of the MLP neural networks of the ground and nonground classifications, which we used to produce the final urban map. Both models show



(a)



(b)

Fig. 15 Summary of LiDAR multilevel classification's best and worst achieved results: F_1-score (test 1 set). (a) Scenario perspective. (b) Class perspective.

good performance on training and validation samples since both reach minimal losses (errors). However, good learning is revealed here by the loss convergence of the training and validation datasets around a relative value each time the epochs increase. On the contrary, overfitting occurs when the validation curve deviates with higher loss values from the training curve after a convergence. In comparison, underfitting happens when training data always show lower loss values than the validation samples. In this case, the validation curve either declines or levels off with the increase in epochs.[55]

Figure 18 shows the classified LiDAR point cloud as the final produced urban thematic map, also used for the qualitative assessment of the classification. The map [Fig. 18(a)] shows the eight classes accurately placed as the quantitative evaluation results suggest, in comparison with the corresponding aerial image [Fig. 18(b)] used in the data registration process. We highlight on the map example locations of five misclassification cases explained as follows:

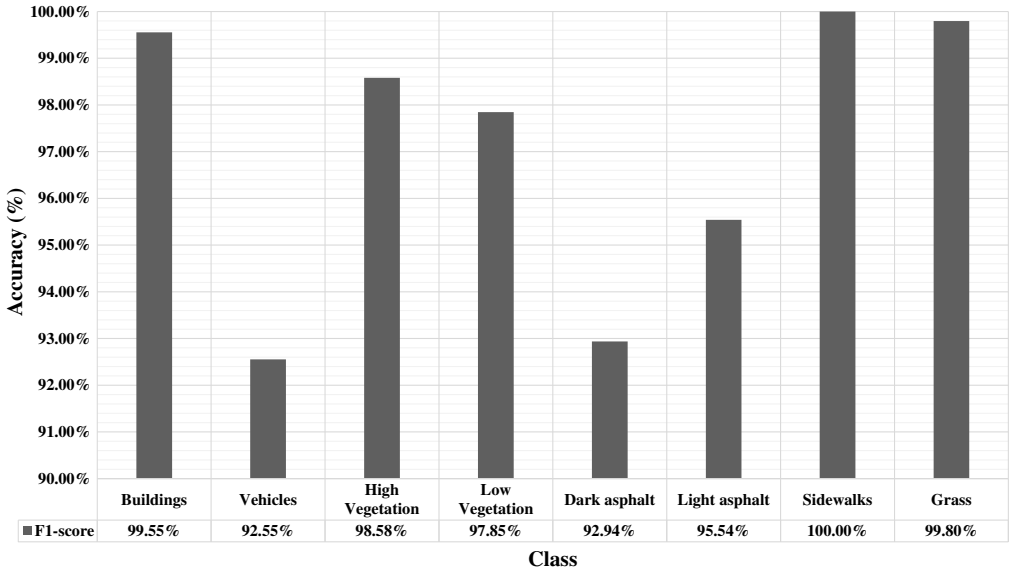


Fig. 16 Produced urban thematic map's per-class accuracies: F_1-score (test 1 set).

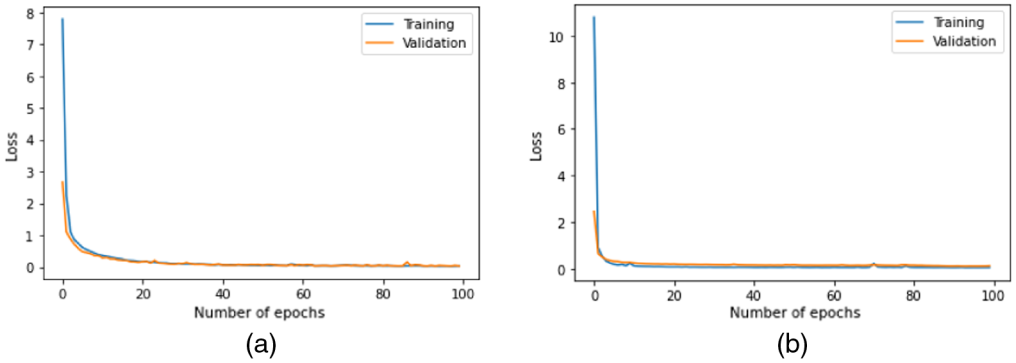
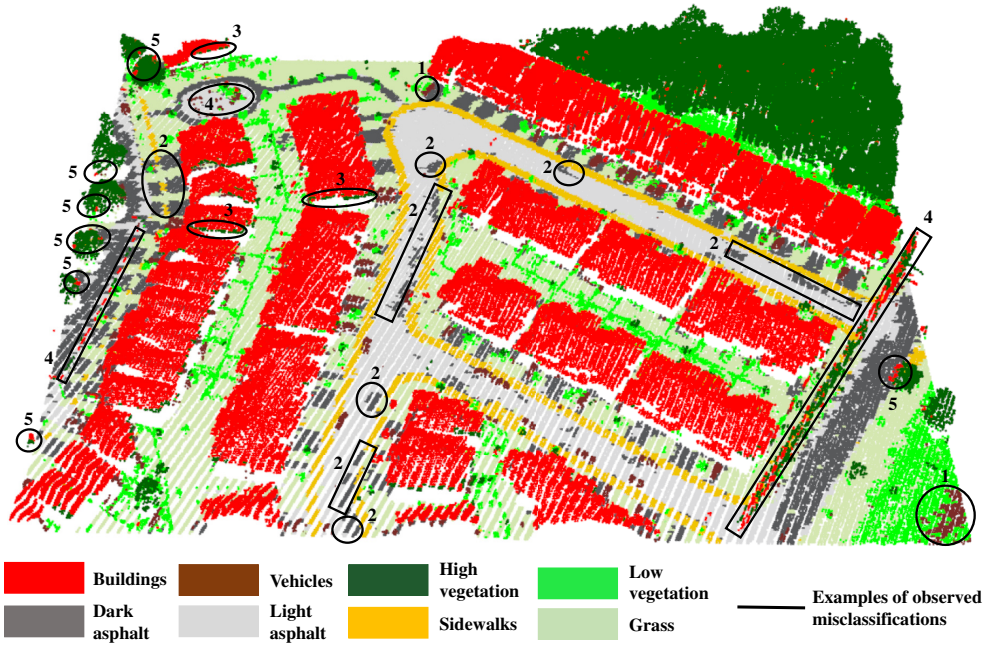
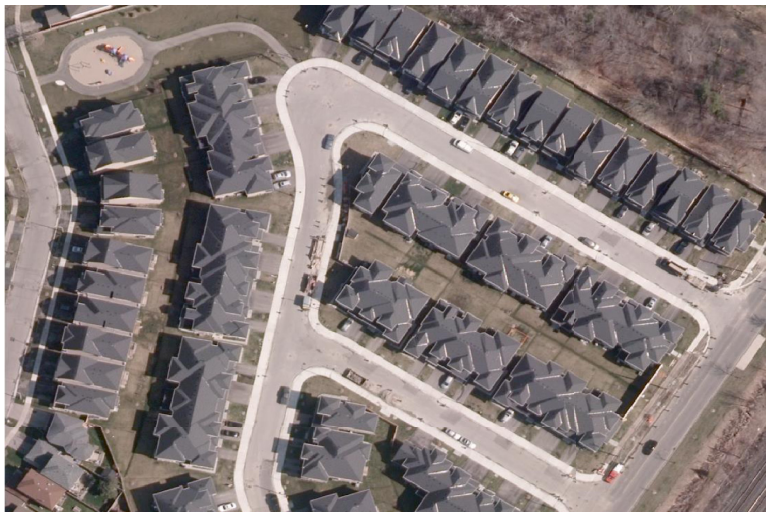


Fig. 17 Produced urban thematic map's MLP learning curves. (a) Nonground model (scenario 9). (b) Ground model (scenario 10).

1. Misclassification of the vehicles and low vegetation classes is still casually spotted even when including geometric indices in the classification feature space. The misclassification affects both classes' accuracy in comparison with the remaining classes (Fig. 16).
2. LiDAR points happen to appear at shadow positions in the corresponding aerial image. They accordingly inherited a spectrum similar to the dark asphalt's, leading to light-/dark-asphalt and grass/dark-asphalt misclassifications. Some of these locations are for asphalt-parking vehicles in the aerial image that did not show up at the acquisition time of the LiDAR data. They also inherited a spectrum close to the dark asphalt's, causing a light-/dark-asphalt misclassification. Both classes appear with lower accuracies relative to the other classes (Fig. 16).
3. Building points are misclassified as high vegetation due to inaccurate aerial photo orthorectification. The misclassification still shows up despite the inclusion of geometric indices in the classification. Figure 16 does not reflect this misclassification, as its locations are not covered within the testing data.
4. The oval mark covers a seesaw in a playground, while the rectangular marks include transmission towers. These objects do not frequently appear in the scene; thus, we did not assign them separate classes. Consequently, the seesaw's segments are labeled low vegetation and vehicles, and the tower segments are classified as buildings and high vegetation.



(a)



(b)

Fig. 18 Produced urban thematic map. (a) Classified point cloud with examples of misclassification locations: (1) vehicles/low-vegetation; (2) shadow and different acquisition time of data; (3) buildings/high-vegetation; (4) objects with no class representation; and (5) buildings/high-vegetation. (b) Corresponding aerial image: R, G, and B bands visualized in RGB colors.[56]

5. High vegetation segments are misclassified as buildings. The ovals mark LiDAR points of trees that were not planted at the acquisition time of the aerial photo. They obtained the spectrum of dark asphalt after registration, which is radiometrically close to building roofs. The height-derived geometric indices solved the problem; however, a few misclassified spots are still located.

Nonetheless, our results based on the qualitative and quantitative assessments outperform the nonground and overall accuracies achieved by Megahed et al.,[35] who carried out a point-based classification with scenario 8 on the same LiDAR point cloud using the same classifier. This comparison underlines the efficiency of introducing height-derived geometric features in classifying urban objects that vary in height values.

5 Conclusions

This study investigates the effect of fusing LiDAR and imagery data on the object-based classification of LiDAR point clouds acquired for urban scenes. A multispectral LiDAR point cloud for a residential area expanded its height and three spectra with R, G, B, and NIR properties from a previous georegistration to an aerial photo covering the same study zone. We filtered ground points from nonground points using the progressive TIN densification approach. Then, we applied the color-based segmentation algorithm on the LiDAR data by calculating the RD between the points based on their R, G, B, and NIR characteristics, in addition to the geometric 3D Euclidian distance. Afterward, we computed geometric and radiometric indices from the LiDAR's height and three channels, besides the R, G, B, and NIR imagery spectra, respectively. We constructed 10 different feature sets representing 10 classification scenarios, some gradually accumulating the geo-registered LiDAR data's spectra to the height feature. The rest of the scenarios accumulated the calculated geometric and radiometric indices (full space), and the last scenario's feature space was the PCA's projection of the full space using the most significant PCs. Subsequently, we collected segments for classification models' training, validation, and testing. Finally, we conducted a supervised object-based classification on the LiDAR point cloud for each considered scenario using MLP neural networks, based on eight observed classes: buildings, vehicles, high vegetation, low vegetation, dark asphalt, light asphalt, sidewalks, and grass. We verified the 10 classification models by two testing sets in segment and point formats, in addition to the k-fold cross-validation.

The models' evaluation on the validation and testing sets and the k-fold approach showed consistent results, indicating reliable models for classifying unseen data. In general, the overall accuracy increased with the gradual expansion of the feature spaces, from ≈80% in a single LiDAR channel scenario to >98% in a full feature space. However, high accuracies were more pronounced in the nonground classification (from ≈90% to >98%) than the ground classification (from ≈61% to >97%). The reason is that the height and height-derived features were not predominant in classifying ground classes, as they insignificantly varied in height values. In contrast, radiometric features were principal in classifying ground objects; consequently, ground accuracy saw a peak in the dual LiDAR channel scenarios (≈85%), followed by another improvement when the inherited aerial photo characteristics were introduced (>97%). Whereas nonground classification also witnessed a peak by including the inherited aerial photo's spectra but less tangibly (>97%). The full feature space marked another peak for the nonground classification (>98%).

Some misclassifications were noticed among classes due to acquiring aerial and LiDAR data separately and shadow and orthorectification issues with the aerial image. Vehicles, dark, and light asphalts were the most problematic classes; nevertheless, they exceeded 90% with the inclusion of the LiDAR and imagery data's spectra and the full feature space.

Buildings were the best-detected class by majority scenarios, starting with a >95% accuracy that grew with the expansion in the feature space. High vegetation and low vegetation were captured with >80% in all scenarios, whose accuracies also rose when input features accumulated. Sidewalks were big hits (>97%) in the single LiDAR channel (C_3), dual LiDAR channels (C_1, C_3), and the PC scenarios. The grass was a remarkable success (>99%) with the inclusion of the LiDAR and imagery data's spectra and the full feature space. Depending on the class accuracy threshold of the mapping application, C_1 may be an option to identify sidewalks (≈88%) and grass (≈81%). C_2 fairly detected grass (≈78%) and light asphalt (≈66%). Likewise, C_3 moderately detected grass (≈72%) and somewhat vehicles (≈64%), with the height included. Dual and triple LiDAR channels can be alternative scenarios for targeting light asphalt (from ≈73% to ≈78%). C_2, and C_3 provided another somewhat vehicle detection (≈68%), with the height included. Introducing the LiDAR and imagery data's spectra granted outstanding overall and per-class accuracies. However, the full feature space better solved misclassified classes, and the projected feature space in the 10th scenario presented the highest ground classification figures. The highest accuracy achieved for vehicles and dark asphalt (>92%) was relatively low and could be enhanced by incorporating hyperspectral features.

We produced the final map applying mixed scenarios: full and projected feature spaces for nonground (98.74%) and ground (97.84%) classifications. The overall mapping accuracy reached 98.43%.

6 Appendix

Table 2 provides the confusion matrixes resulting from the segments and points validation datasets; Test 1 and Test 2, respectively. They show the per-class and overall accuracies of each

Table 2 Confusion matrixes (ground truth: rows; predictions: columns) of testing analysis for all classification scenarios. (Class 1: buildings, class 2: vehicles, class 3: high vegetation, class 4: low vegetation, class 5: dark asphalt, class 6: light asphalt, class 7: sidewalks, class 8: grass).

		Testing datasets																
		Test 1									Test 2								
Scenario		1	2	3	4	5	6	7	8	F_1-score (%)	1	2	3	4	5	6	7	8	F_1-score (%)
Scenario 1	1	1100	0	16	5					95.16	1770	0	18	1					95.42
	2	0	32	0	63					47.76	0	45	0	83					50.00
	3	80	0	204	0			ϕ		80.95	146	0	347	0			ϕ		80.89
	4	11	7	0	304					87.61	5	7	0	376					88.68
	5					71	0	0	68	31.14					141	0	0	230	20.66
	6					191	0	0	39	0.00					794	0	0	113	0.00
	7			ϕ		0	0	98	9	88.29			ϕ		0	0	291	24	85.09
	8					55	0	17	424	81.85					59	0	78	809	76.25
		Overall accuracy (%) = 79.92									Overall accuracy (%) = 70.81								
Scenario 2	1	1085	0	28	8					95.89	1735	0	47	7					96.52
	2	0	0	0	95					0.00	0	0	0	128					0.00
	3	53	0	231	0			ϕ		85.08	69	0	423	1			ϕ		87.85
	4	4	0	0	318					85.60	2	0	0	386					84.84
	5					0	115	0	24	0.00					0	273	0	98	0.00
	6					0	214	0	16	66.56					0	842	0	65	77.93
	7			ϕ		0	0	0	107	0.00			ϕ		0	0	0	315	0.00
	8					0	84	0	412	78.10					0	139	0	807	72.34
		Overall accuracy (%) = 80.89									Overall accuracy (%) = 87.56								
Scenario 3	1	1098	0	17	6					96.40	1771	0	16	2					97.23
	2	0	51	0	44					64.15	0	71	0	57					63.11
	3	58	0	226	0			ϕ		85.77	82	1	408	2			ϕ		88.99
	4	1	13	0	308					90.59	1	25	0	362					89.27
	5					0	0	0	139	0.00					0	0	0	371	0.00
	6					0	0	0	230	0.00					0	0	0	907	0.00
	7			ϕ		2	0	103	2	98.10			ϕ		1	0	311	3	98.57
	8					0	0	0	496	72.78					4	0	5	937	59.23
		Overall accuracy (%) = 81.68									Overall accuracy (%) = 72.33								

Table 2 (*Continued*).

		Testing datasets																
		Test 1									Test 2								
Scenario		1	2	3	4	5	6	7	8	F_1-score (%)	1	2	3	4	5	6	7	8	F_1-score (%)
Scenario 1 4	1	1091	0	26	4					96.63	1731	0	54	4					96.38
	2	0	34	0	61					45.33	0	45	0	83					46.39
	3	43	0	241	0			ϕ		87.48	70	0	422	1			ϕ		87.10
	4	3	21	0	298					87.01	2	21	0	365					86.80
	5					2	119	0	18	2.84					0	338	0	33	0.00
	6					0	225	0	5	73.89					0	900	0	7	81.97
	7			ϕ		0	2	101	4	92.24			ϕ		0	7	304	4	97.12
	8					0	33	11	452	92.72					3	44	7	892	94.79
		Overall accuracy (%) = 87.47									Overall accuracy (%) = 87.30								
Scenario 1 5	1	1090	4	21	6					97.41	1753	0	30	6					97.80
	2	0	44	0	51					59.06	0	63	0	65					64.62
	3	23	0	261	0			ϕ		92.06	42	0	451	0			ϕ		92.61
	4	4	6	1	311					90.14	1	4	0	383					90.97
	5					26	102	1	10	30.77					49	306	0	16	23.06
	6					2	222	0	6	76.16					2	898	0	7	84.32
	7			ϕ		0	4	103	0	97.63			ϕ		0	5	310	0	98.73
	8					2	25	0	469	95.62					3	14	3	926	97.73
		Overall accuracy (%) = 90.41									Overall accuracy (%) = 90.56								
Scenario 1 6	1	1104	0	12	5					98.18	1770	0	18	1					98.39
	2	0	61	0	34					68.16	1	73	0	54					64.32
	3	22	0	262	0			ϕ		93.57	36	0	456	1			ϕ		94.31
	4	2	23	2	295					89.94	2	26	0	360					89.55
	5					28	103	0	8	29.47					70	279	0	22	28.99
	6					18	208	0	4	76.47					35	866	0	6	84.08
	7			ϕ		3	0	104	0	97.65			ϕ		2	0	312	1	99.05
	8					2	3	2	489	98.09					5	8	3	930	97.64
		Overall accuracy (%) = 91.30									Overall accuracy (%) = 90.63								

Table 2 (*Continued*).

		Testing datasets																
		Test 1									Test 2							
Scenario		1	2	3	4	5	6	7	8	F_1-score (%)	1	2	3	4	5	6	7	8	F_1-score (%)
Scenario 17	1	1100	1	15	5					97.86	1770	0	15	4					98.44
	2	4	41	0	50					58.99	2	53	0	73					57.30
	3	19	0	265	0			ϕ		93.97	33	0	460	0			ϕ		95.04
	4	4	2	0	316					91.20	2	4	0	382					90.20
	5					31	88	0	20	33.33					50	275	0	46	22.27
	6					12	212	0	6	78.81					26	872	1	8	84.41
	7			ϕ		1	2	101	3	97.12			ϕ		0	3	311	1	99.04
	8					3	6	0	487	96.25					2	9	1	934	96.54
		Overall accuracy (%) = 91.37									Overall accuracy (%) = 90.54								
Scenario 18	1	1108	3	6	4					98.80	1781	0	8	0					99.25
	2	2	83	0	10					88.30	0	118	0	10					87.41
	3	11	0	273	0			ϕ		96.98	14	0	478	1			ϕ		97.65
	4	1	7	0	314					96.62	5	24	0	359					94.72
	5					129	7	0	3	90.85					344	23	0	4	92.97
	6					15	215	0	0	95.13					24	883	0	0	97.41
	7			ϕ		0	0	107	0	100.00			ϕ		0	0	315	0	100.00
	8					1	0	0	495	99.60					1	0	0	945	99.74
		Overall accuracy (%) = 97.49									Overall accuracy (%) = 97.86								
Scenario 19	1	1116	2	1	2					99.55	1787	2	0	0					99.83
	2	0	87	1	7					92.55	1	117	1	9					84.78
	3	5	0	278	1			ϕ		98.58	0	0	491	2			ϕ		99.70
	4	0	4	0	318					97.85	3	29	0	356					94.30
	5					122	17	0	0	91.39					365	4	0	2	87.74
	6					5	223	2	0	94.29					92	815	0	0	94.27
	7			ϕ		0	3	104	0	97.65			ϕ		0	3	312	0	99.52
	8					1	0	0	495	99.90					4	0	0	942	99.68
		Overall accuracy (%) = 98.17									Overall accuracy (%) = 97.15								

Table 2 (*Continued*).

		Testing datasets																
		Test 1									Test 2							
Scenario		1	2	3	4	5	6	7	8	F_1-score (%)	1	2	3	4	5	6	7	8	F_1-score (%)
Scenario 10	1	1108	3	2	8					99.02	1782	3	2	2					99.47
	2	6	77	0	12					87.01	8	110	0	10					90.16
	3	2	0	281	1			ϕ		98.94	1	0	491	1			ϕ		99.59
	4	1	2	1	318					96.22	3	3	0	382					97.57
	5					125	14	0	0	92.94					338	31	0	2	93.89
	6					5	22	0	0	95.54					11	896	0	0	97.55
	7			ϕ		0	0	107	0	100.00			ϕ		0	2	313	0	99.68
	8					0	2	0	494	99.80					0	1	0	945	99.84
		Overall accuracy (%) = 97.89									Overall accuracy (%) = 98.50								
Final map	1	1116	2	1	2					99.55	1787	2	0	0					99.83
	2	0	87	1	7					92.55	1	117	1	9					84.78
	3	5	0	278	1			ϕ		98.58	0	0	491	2			ϕ		99.70
	4	0	4	0	318					97.85	3	29	0	356					94.30
	5					125	14	0	0	92.94					338	31	0	2	93.89
	6					5	22	0	0	95.54					11	896	0	0	97.55
	7			ϕ		0	0	107	0	100.00			ϕ		0	2	313	0	99.68
	8					0	2	0	494	99.80					0	1	0	945	99.84
		Overall accuracy (%) = 98.43									Overall accuracy (%) = 98.24								

classification scenario. Both validation datasets produce comparable accuracy figures, which reflects the consistency of the designed classification models and their reliability to predict unseen data.

Acknowledgments

This research was funded by the Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) (RGPIN-2015-03960), the FCE Start-up Fund of the Hong Kong Polytechnic University (BE2U), and the Early Career Scheme (Project Number: 25213320) by the Research Grants Council of the Hong Kong Special Administrative Region. The authors would also like to thank Dr. Ernest Ho for his contribution in proofreading the paper.

References

1. K. Krishnaveni and P. Anilkumar, "Managing urban sprawl using remote sensing and GIS," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **XLII-3/W11**, 59–66 (2020).
2. M. Steurer and C. Bayr, "Measuring urban sprawl using land use data," *Land Use Policy* **97**, 104799 (2020).

3. United Nations, Department of Economic and Social Affairs. Population Division, *World Urbanization Prospects – The 2018 Revision* (2019).
4. M. Acuto et al., "Seeing COVID-19 through an urban lens," *Nat. Sustainability* **3**(12), 977–978 (2020).
5. S. D. Whitaker, "Did the COVID-19 pandemic cause an urban exodus?" Cfed District Data Briefs (cfddb 20210205) (2021).
6. J. A. Leech et al., "It's about time: a comparison of Canadian and American time–activity patterns," *J. Exposure Sci. Environ. Epidemiol.* **12**(6), 427–432 (2002).
7. T. Peters and A. Halleran, "How our homes impact our health: using a COVID-19 informed approach to examine urban apartment housing," *Archnet-IJAR* **15**, 10–27 (2020).
8. L. Guo et al., "Relevance of airborne LiDAR and multispectral image data for urban scene classification using random forests," *ISPRS J. Photogramm. Remote Sens.* **66**(1), 56–66 (2011).
9. T. Long et al., "A generic framework for image rectification using multiple types of feature," *ISPRS J. Photogramm. Remote Sens.* **102**, 161–171 (2015).
10. R. Huang et al., "Semantic labeling and refinement of LiDAR point clouds using deep neural network in urban areas," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **IV-2/W7**, 63–70 (2019).
11. A. Sen, B. Suleymanoglu, and M. Soycan, "Unsupervised extraction of urban features from airborne LiDAR data by using self-organizing maps," *Surv. Rev.* **52**(371), 150–158 (2020).
12. Z. Kang, J. Yang, and R. Zhong, "A Bayesian-network-based classification method integrating airborne LiDAR data with optical images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **10**(4), 1651–1661 (2017).
13. S. Sanlang et al., "Integrating aerial LiDAR and very-high-resolution images for urban functional zone mapping," *Remote Sens.* **13**(13), 2573 (2021).
14. F. Rodríguez-Puerta et al., "Comparison of machine learning algorithms for wildland-urban interface fuelbreak planning integrating ALS and UAV-borne LiDAR data and multispectral images," *Drones* **4**(2), 21 (2020).
15. R. Pu and S. Landry, "Mapping urban tree species by integrating multi-seasonal high resolution pléiades satellite imagery with airborne LiDAR data," *Urban For. Urban Greening* **53**, 126675 (2020).
16. Y. Zhang and Z. Shao, "Assessing of urban vegetation biomass in combination with LiDAR and high-resolution remote sensing images," *Int. J. Remote Sens.* **42**(3), 964–985 (2021).
17. Y. He et al., "Integration of InSAR and LiDAR technologies for a detailed urban subsidence and hazard assessment in Shenzhen, China," *Remote Sens.* **13**(12), 2366 (2021).
18. Q. Zhan, Y. Liang, and Y. Xiao, "Color-based segmentation of point clouds," *Laser Scanning* **38**(3), 155–161 (2009).
19. Y. Megahed, A. Shaker, and W. Y. Yan, "A phase-congruency-based scene abstraction approach for 2D-3D registration of aerial optical and LiDAR images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **14**, 964–981 (2021).
20. A. M. Ramiya, R. R. Nidamanuri, and R. Krishnan, "Object-oriented semantic labelling of spectral–spatial LiDAR point cloud for urban land cover classification and buildings detection," *Geocarto Int.* **31**(2), 121–139 (2016).
21. H. Nakawala, G. Ferrigno, and E. De Momi, "Toward a knowledge-driven context-aware system for surgical assistance," *J. Med. Rob. Res.* **2**(3), 1740007 (2017).
22. G. Sanderson, "3Blue1Brown Channel," 3Blue1Brown Channel Website, https://www.3blue1brown.com/ (accessed 23 June 2021).
23. G. Sanderson, "Eigenvectors and eigenvalues – chapter 14: Essence of linear algebra," YouTube Website, https://www.youtube.com/watch?v=PFDu9oVAE-g (accessed 23 June 2021).
24. V. Spruyt, "A geometric interpretation of the covariance matrix," Computer Vision for Dummies Website, https://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/ (accessed 23 June 2021).
25. H. Abdullatif, "Dimensionality reduction for dummies—part 3: connect the dots," Towards Data Science Website, https://towardsdatascience.com/dimensionality-reduction-for-dummies-part-3-f25729f74c0a (accessed 28 June 2021).

26. J. Brownlee, "How to choose a feature selection method for machine learning," Machine Learning Mastery Website, https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/ (accessed 14 July 2021).
27. J. Brownlee, "Introduction to dimensionality reduction for machine learning," Machine Learning Mastery Website, https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/ (accessed 14 July 2021).
28. J. Brownlee, *Basics of Linear Algebra for Machine Learning*, Machine Learning Mastery (2018).
29. J. Brownlee, "What is deep learning?" Machine Learning Mastery Website, https://machinelearningmastery.com/what-is-deep-learning/ (accessed 21 July 2021).
30. J. Brownlee, *Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow using Keras*, Machine Learning Mastery (2016).
31. A. Sharma, "Understanding activation functions in deep learning," Learn OpenCV Website, https://learnopencv.com/understanding-activation-functions-in-deep-learning/ (accessed 21 July 2021).
32. K. Vu, "Activation functions and optimizers for deep learning models," DZone Website, https://dzone.com/articles/activation-functions-and-optimizers-for-deep-learn (accessed 21 July 2021).
33. J. Brownlee, "Difference between a batch and an epoch in a neural network," Machine Learning Mastery Website, https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ (accessed 30 October 2020).
34. Y. Megahed, W. Y. Yan, and A. Shaker, "Semi-automatic approach for optical and LiDAR data integration using phase congruency model at multiple resolutions," *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **XLIII-B3-2020**, 611–618 (2020).
35. Y. Megahed, A. Shaker, and W. Y. Yan, "Fusion of airborne LiDAR point clouds and aerial images for heterogeneous land-use urban mapping," *Remote Sens.* **13**(4), 814 (2021).
36. LAStools, "Lasground-New," Rapidlasso GmbH, https://rapidlasso.com/lastools/lasground (accessed 1 October 2020).
37. P. Axelsson, "DEM generation from laser scanner data using adaptive TIN models," *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **33**, 110–117 (2000).
38. O. Harrison, "Machine learning basics with the k-nearest neighbors algorithm," Towards Data Science Website, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (accessed 17 May 2021).
39. N. Bhatia, "Survey of nearest neighbor techniques," arXiv:1007.0085 (2010).
40. R. F. Sproull, "Refinements to nearest-neighbor searching in k-dimensional trees," *Algorithmica* **6**(1), 579–589 (1991).
41. Algokodabra, "K-D Tree: build and search for the nearest neighbor," YouTube Website, https://www.youtube.com/watch?v=ivdmGcZo6U8 (accessed 18 May 2021).
42. The SciPy Community, "scipy.spatial.cKDTree," SciPy. Org Website, https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.cKDTree.html (accessed 18 May 2021).
43. S. Maneewongvatana and D. M. Mount, "Analysis of approximate nearest neighbor searching with clustered point sets," in *Data Structures, Near Neighbor Searches, and Methodology*, Vol. 59, pp. 105–123 (2002).
44. ENVI – Environment for Visualizing Images, "Broadband greenness," L3 Harris Geospatial Documentation Center Website, https://www.l3harrisgeospatial.com/docs/BroadbandGreenness.html (accessed 29 June 2021).
45. P. Zhang et al., "A strategy of rapid extraction of built-up area using multi-seasonal landsat-8 thermal infrared band 10 images," *Remote Sens.* **9**(11), 1126 (2017).
46. M. Weinmann, B. Jutzi, and C. Mallet, "Feature relevance assessment for the semantic interpretation of 3D point cloud data," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **5**, II-5/W2 (2013).
47. Scikit-Learn Developers, "sklearn.decomposition.PCA," Scikit Learn Website, https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html (accessed 15 July 2021).

48. J. Brownlee, "Ordinal and one-hot encodings for categorical data," Machine Learning Mastery Website, https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/ (accessed 30 October 2020).
49. Keras, "About Keras," Keras Website, https://keras.io/about/ (accessed 22 July 2021).
50. Keras, "Adam," Keras Website, https://keras.io/api/optimizers/adam/ (accessed 1 November 2020).
51. Keras, "Probabilistic losses," Keras Website, https://keras.io/api/losses/probabilistic_losses/ (accessed 1 November 2020).
52. Keras, "Layer activation functions," Keras Website, https://keras.io/api/layers/activations/#relu-function (accessed 22 July 2021).
53. J. Brownlee, *Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-to-End*, Machine Learning Mastery (2016).
54. J. Brownlee, "Why do I get different results each time in machine learning?" Machine Learning Mastery Website, https://machinelearningmastery.com/different-results-each-time-in-machine-learning/ (accessed 4 August 2021).
55. J. Brownlee, "How to use learning curves to diagnose machine learning model performance," Machine Learning Mastery Website, https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/ (accessed 21 September 2021).
56. Ontario Ministry of Natural Resources, "Greater Toronto Area (GTA) orthophotography project 2013," Peterborough Region, http://geo2.scholarsportal.info.ezproxy.lib.ryerson.ca (accessed 8 October 2020).

Yasmine Megahed received her MSc degree in geospatial technologies with a major in remote sensing from the NOVA IMS Information Management School, Lisbon, Portugal, in 2015. She is currently working toward her PhD with the Department of Civil Engineering, Ryerson University, Toronto, Ontario, Canada. Her research focuses on remote sensing applications, especially digital urban mapping that integrates LiDAR and imagery data in the point cloud classification of urban morphologies.

Wai Yeung Yan received his PhD in civil engineering from Ryerson University, Toronto, ON, Canada, in 2012. He is currently an assistant professor with the Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, and an adjunct professor with the Department of Civil Engineering, Ryerson University. His research interests include point cloud processing, laser scanning, and remote sensing.

Ahmed Shaker received his PhD in satellite sensor modeling from the Department of Land Surveying and Geo-Informatics, the Hong Kong Polytechnic University, Hong Kong, in 2004. He is currently a professor with the Department of Civil Engineering and an associate dean of the Faculty of Engineering and Architecture Science, Ryerson University, Toronto, Ontario, Canada. He holds two patents and has more than 130 publications in international journals and conferences. His research interests include LiDAR data processing, satellite sensor modeling, image segmentation and classification, and 3-D modeling. He was the recipient of a number of national and international awards. He is currently serving as the president of the Canadian Remote Sensing Society (2020–2022).