2D Building Footprint Extraction from LiDAR

Sarvesh Gharat[a], Sona Elza Simon[a], Apurva Dhingra[a], Deekshant Kumar[a], Sandhya Dasari[b]

[a]*Center for Machine Intelligence and Data Science, IIT Bombay, Mumbai, India*
[b]*Department of Civil Engineering, IIT Bombay, Mumbai, India*

Abstract

In this manuscript, aerial LiDAR points are used to extract 2D footprints and are visualized on the base map. Various Machine Learning algorithms were experimented to extract the building points from the raw data. Different clustering algorithms like K-Means and DBSCAN were explored to convert LiDAR points into optimal building clusters. Algorithms like Convex Hull and Alpha Shape are explored to extract the footprints by converting the clusters into polygons.

Keywords: `Machine Learning, Neural Network, LiDAR, Classification, Clustering`

1. Introduction

LiDAR (Light Detection and Ranging) is a technology that provides 2D/3D terrain mapping through laser ranging and scanning. LiDAR data points can be obtained from either from aerial or terrestrial views. This data is collected as a 'data cloud' of each point that gets reflected from the surface of everything around it, including building structures and vegetation. These extra structures are removed when LiDAR is used to develop Digital Elevation models (DEM) [1]. LiDAR data points are used for mapping an area's topography, agriculture, archaeology, and autonomous driving [2], among others. Extraction of the automatic building footprint, or the digital representation of the building's border [3], is an additional necessity for Geographic information systems (GIS) for urban planning or disaster management. Current methods of extracting footprints rely heavily on pre-existing building footprints for modeling [4], instead of purely machine learning techniques.

In this study, we explore aerial data points to identify their footprints, i.e. using one of the methods from three categories of building extraction methods [5]. The three methods are i) high-definition aerial imagery with pre-existing knowledge for building extraction modeling, ii) using only data-driven machine learning algorithm for LiDAR points or iii) a combination of the two. The approach of using LiDAR data points is better than using high-resolution aerial imagery, which is generally not reliable enough for practical use. Secondly, due to improvements in LiDAR sampling technology, relying on manual footprint modeling will be slow and inefficient for such an expensive resource. Therefore, it is necessary to investigate ways to detect building footprints using automated procedures and machine learning approaches.

In this study, 2D building footprint extraction is done from aerial LiDAR point clouds and is visually evaluated on open street maps. Our code is made available on GitHub.[1]

[1] `https://github.com/DeekshantKumar/CS725---project`

2. Preliminaries

2.1. GAN

A quick review of what is a GAN (General adversarial network): GANs are generative models: they create new data instances that resemble your training data by training a general noise following a certain probability distribution. For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real or fake (generated). The two models are trained together in a zero-sum game, adversarial until the discriminator model is fooled about half the time. The model architecture for GAN is shown in Figure 1.

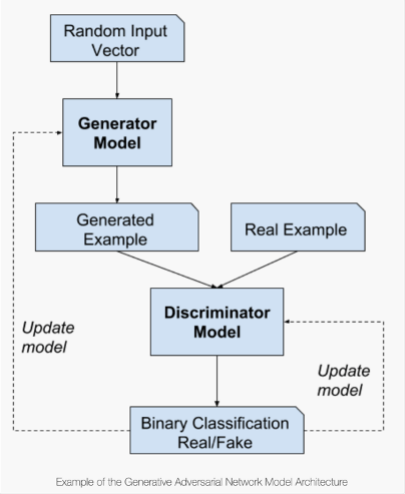

Figure 1: Model architecture of GAN.Reference [6]

2.2. K-Means

K-Means is one of the most popular clustering algorithms that clusters data points based on their closeness to the centroid. It tries to minimize the squared euclidean distance of each point in a cluster from its cluster centroid. The k-means clustering algorithm takes two steps iteratively: 1) fixing the cluster centroids, it associates each point with one of the k clusters, by minimizing the squared euclidean distance, and 2) using the points associated with a cluster, finding the mean value of the centroid of each cluster. These two steps are repeated until convergence [7].

2.3. DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [8] clusters points based on the density of points. It uses two parameters: the minimum number of points to specify a dense region, min_pts, and a radius measure to locate points in its neighborhood, ϵ. Basic terminology used in this algorithm includes core points, border points, and noise as shown in Figure 2. The core point is any point that is surrounded by at least min_pts (say 4 including itself) number of points within the euclidean distance of ϵ. The border point is the point that at least has one core point within distance ϵ. Noise is the point that has less than min_pts in its neighborhood and isn't connected to a core point, unlike the border point.

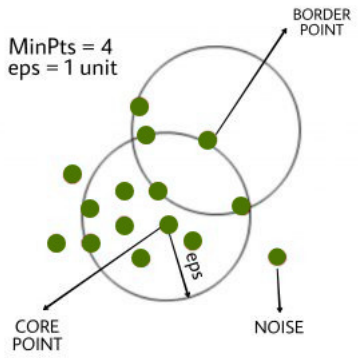

Figure 2: DBSCAN algorithmic representation[9]

The algorithm for DBSCAN clustering is as follows :

1. Pick a point from the dataset arbitrarily (ensuring all points are visited).
2. Check if there are min_pts within the neighborhood of the point within the radius of ϵ, then all those points belong to the same cluster.
3. If a core point isn't already part of a cluster, form a new one. Find all the densely connected points and associate them to the same cluster having a core point.
4. Iterate through all the unvisited points in the dataset and repeat the above sets.
5. The points that do not belong to any cluster are marked as noise or outliers

2.4. Convex Hull

Given a set of points in the plane, the convex hull of the set is the smallest convex polygon that contains all the given points. The convex hull for a given set of points is shown in Figure 3. In this work, we explore the Quick-hull Algorithm for the implementation of the Convex Hull. The Quick-hull algorithm [10] can be broken down into the following steps:

1. Find the points with minimum(min) and maximum(max) x coordinates. If many points with the same min/max x exist, use ones with min/max y correspondingly.
2. Use the line formed by the two points to divide the set into two subsets of points.
3. Determine the point, on one side of the line, with the max distance from the line. This point forms a triangle with those of the line.
4. Ignore the points lying inside of that triangle.
5. Repeat steps 2 and 3 on the new lines formed by the triangle until no more points are left.

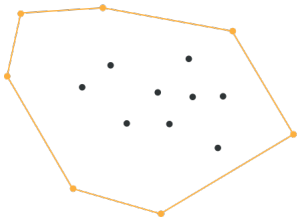

Figure 3: An example of Convex Hull for a given set of points [10].

2.5. Alpha Shape

Given a set of points, the Alpha Shape algorithm helps to generalize bounding polygons containing the given set of points by defining an alpha parameter(α). The algorithm draws disks of radius $1/\alpha$ for the given set of points and only those edges or triangles of the Delaunay Triangulation [11] whose disks meet each other are considered to form alpha complexes. The alpha shape gives a convex hull when $\alpha = 0$. The Alpha Shape structure for a set of points at a given α value is shown in Figure 4.

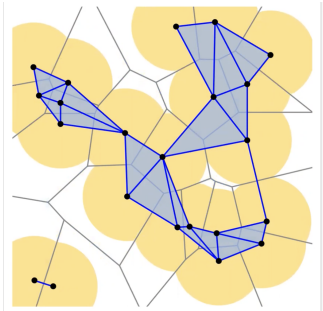

Figure 4: An example of the Alpha Shape algorithm for a given set of points and α value [12].

2.6. t-SNE

t-Distributed Stochastic Neighbor Embedding(t-SNE) is a dimensionality reduction technique that outputs the relevant features. The key aspect of the t-SNE model is its ability to preserve local details. This means, roughly, the points that are close to one another in the high-dimensional data set will tend to be close to one another in the chart. An added advantage of t-SNE is the beautiful visualizations of 2D or 3D plots along with some additional python libraries such as seaborn.

We can do also use EDA techniques to understand feature space in data and derive some insights, but nothing beats a chart. However, fitting multiple dimensions of data into a simple chart is always a challenge (dimensionality reduction). This is where t-SNE (or, t-distributed stochastic neighbor embedding) becomes helpful. Figure 5 shows the working of t-SNE.

The algorithm can be broken down into the following steps:

1. Find the similarity between 2 points which is done by placing all the points on a normal distribution (refer fig. 6) curve and estimating their distance to create a similarity matrix.
2. Repeat step 1 by using t-distribution (refer fig. 7) instead of normal distribution for measuring the similarity.
3. Optimize this similarity matrix as close as possible to the original similarity matrix, this is done using the process of an algorithm trying to minimize the Kullback–Leibler divergence (KL divergence) through gradient descent.

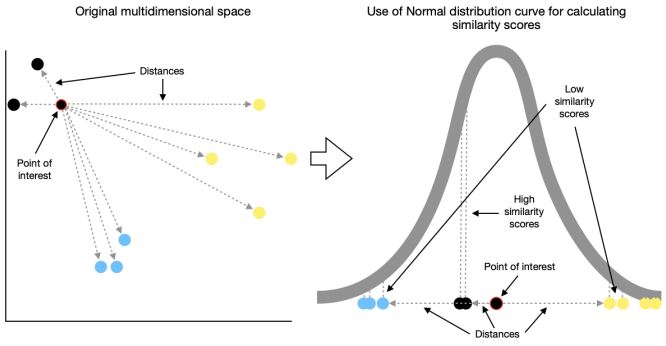

Figure 5: The concept diagram of t-SNE.Reference [6]

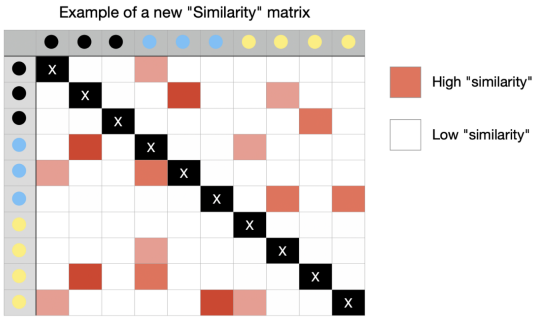

Figure 6: Similarity matrix using normal distribution. Reference [6]

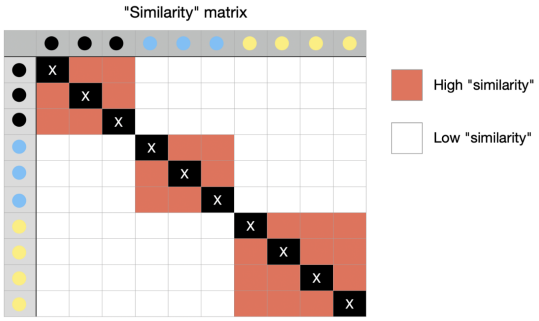

Figure 7: Similarity matrix using t-distribution.Reference [6]

3. Methodology

3.1. Overall Architecture

The aerial LiDAR data points are pre-processed using ”Coordinate Reference System (CRS)” [13]. Further, the point clouds are classified into different classes using various supervised classification techniques. The point clouds corresponding to the ”building” class are further processed to obtain the individual building using clustering algorithms like - KMeans, DBSCAN [8] followed by finding the cluster/ building boundary using Convex Hull, Alpha Shape [14]. We also explored the use of GAN (General Adversarial Network)[15] to strengthen the data by generating more point clouds. The building footprints are evaluated by visualizing them on Open Street Maps. The overall architecture is shown in Figure 8.

3.2. Data Pre-processing

The pre-processing step involves using techniques for data projection using existing libraries. The dataset contains the following features: x, y, z coordinates, return number of the laser pulse, class, and intensity. We made use of the GeoPanda library to convert 3D coordinates into a geographic 2D flattened representation (that can be visualized on a map) using Coordinate Reference System (CRS). Many GIS/map software does support the rectangle-type flattened representation of data which is the transformation done to the (x,y) coordinates. Raw LiDAR data are visualized on the map using the Folio python library and Kepler-GL library as shown in the images.

3.3. EDA

A couple of exploratory data analysis techniques were chosen to understand feature independence such as correlation matrix and t-SNE as described above. As it is seen from the correlation matrix (Fig. 10), variables are independent, hence it can be used for further analysis. It was also plotted in 3D as shown in (Figure 13). However, the next step is to see if the variables can be separately linearly, variables are well separated in 2D/3D space using the t-SNE method. The LiDAR dataset was then plotted in 2D space (refer Fig. 9) to see if it is visually tedious to extract any key indication, however when the data was plotted in 3D space (refer Fig. 12), it can be seen some of the classes became separable, but still, it was not completely

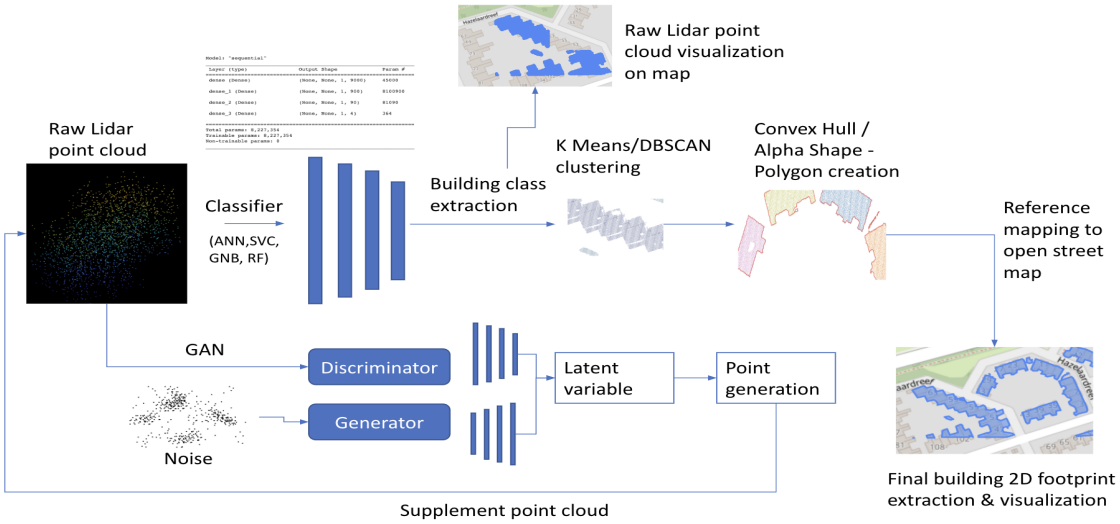

Figure 8: Overall Architecture.

linearly separable, with this further processing of classification was conducted

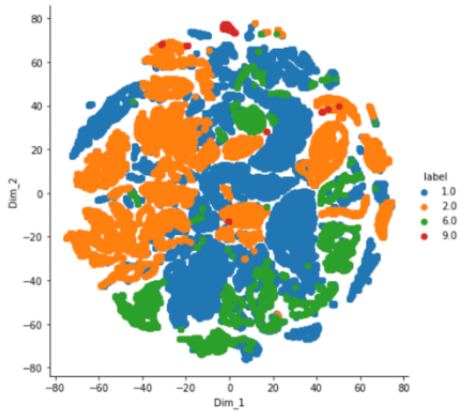

Figure 9: 2D t-SNE plot from LiDAR data.

3.4. GAN

In the GAN experiment, the LiDAR points were fed into a CNN network to discriminate the generated points from the generator network using the same CNN model. The Vanilla GAN model was used for training (noise from a random normal distribution was used) to train latent variables, in order to generate more number of points, though it is seen from a generator and discriminator model, loss (refer fig. 14) converges at a loss %, it did not turn out to be helpful in creating additional synthetic data for LiDAR point cloud.

3.5. Classification

The initial part of the study focuses on using multiple classification techniques to identify the building class. This is done to

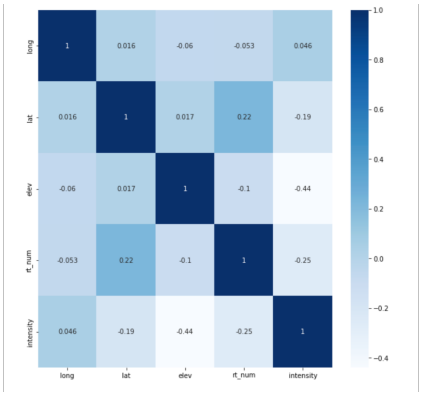

Figure 10: Correlation Matrix on different independent variables.

ensure that the algorithm can identify the correct points to proceed within the cases wherein there are no labeled data available. Although this is limited to regions with similar attributes yet it plays an important role in this study.

Before coming up with an optimal model, a detailed comparison was made between traditional Machine [16][17] [18] [19] and Deep Learning models[20].

As seen in Figure 11, Gaussian Naive Bayes doesn't perform well compared to other algorithms. This is because of the conditional dependency which Naive Bayes takes into consideration. Further, as the data isn't linearly separable (completely), the traditional ML algorithms do not perform well as compared to ANN. Although Random forest due to boosting gives similar accuracy as that of ANN, but the class-specific F1-Score of the same is comparatively lesser than that of ANN. Hence, the optimal classifier for this particular happens to be a Neural Network model.

3.6. Clustering

We explored both k-means and DBSCAN for clustering the buildings from the classified data of building class. In k-means,

Models		F1-Score			
	Accuracy	Class=1	Class=2	Class=6 (Building)	Class=9
ANN	98%	0.97	0.98	0.97	0.84
SVC	94%	0.93	0.96	0.79	0.84
Decision Tree	96%	0.96	0.97	0.92	0.91
Random Forest	98%	0.98	0.98	0.94	0.92
Gaussian NB	86%	0.83	0.95	0.56	0.78

Figure 11: Detailed comparison of different classifiers.

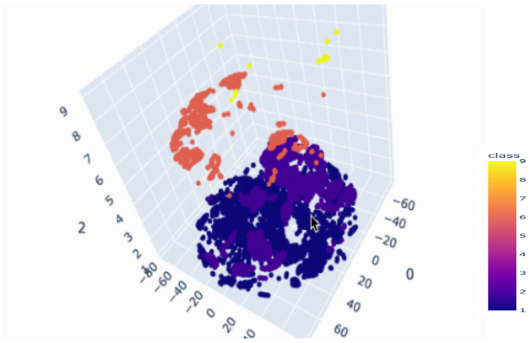

Figure 12: 3D t-SNE plot from LiDAR data.

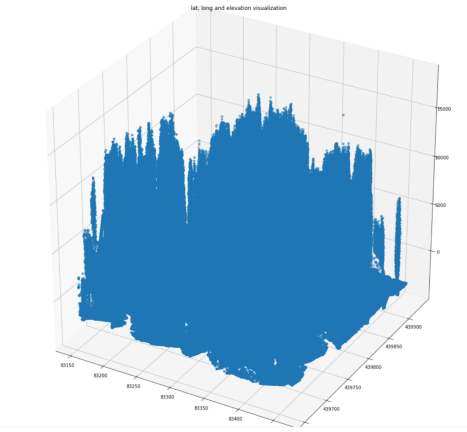

Figure 13: 3D plot of elevation and geodata(lat/long).

the number of clusters is obtained by observing the visualization of raw LiDAR points on open street maps using the Folio library and interactive Klepner GL. Using this value of k = 18, the k-means clustering algorithm is implemented on the LiDAR data points as shown in Figure 15. Results are visualized on a map again to identify their correctness. It is observed that k-means clustering has overdone the clustering for one of the connected buildings, giving inappropriate building cluster boundaries.

Given the nature of LiDAR data points, a density-based algorithm will be a better choice. So, DBSCAN was experimented with as well. In DBSCAN, using hyperparameter fitting over a range of values for min_pts, ϵ, the following values 4 and 0.5 were finally used respectively. We found that our data has 18

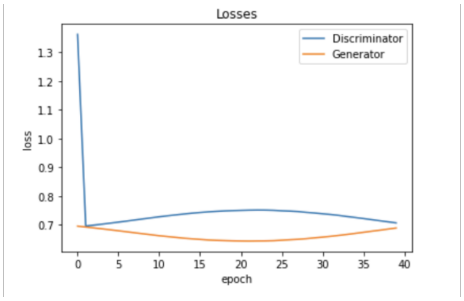

Figure 14: Discriminator and Generator loss curve.

building clusters as depicted in Figure 16. If we compare this result with satellite imagery as laid out on the Open Street map, Figure 19, we observe that closely connected buildings have not been separated by this algorithm. Despite this, DBSCAN gave a good approximation of building clusters.

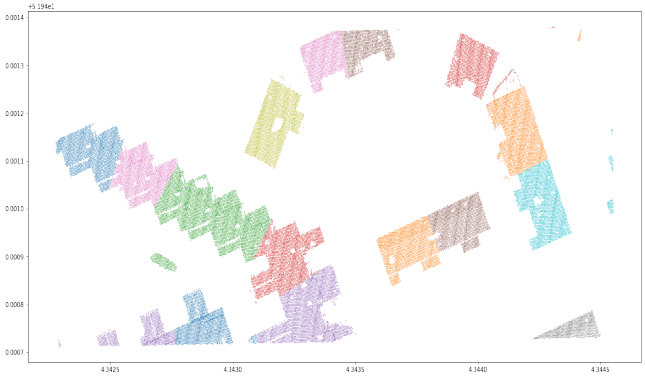

Figure 15: Clustering obtained using K-means algorithm.

3.7. *Footprint Extraction*

We explored both Convex Hull and Alpha Shape algorithms for the extraction of building footprints from the clusters obtained from DBSCAN. The footprints obtained using the Convex Hull are shown in Figure 17. We can clearly observe that the algorithm failed to obtain a good boundary approximation for the given cluster of points. This is because the building clusters generally have non-convex structures while Convex Hull tries to obtain a convex boundary for the given set of points.

To handle the non-convex nature of the clusters, we explored the Alpha Shape algorithm which provides a generalized algorithm for boundary polygons. The footprints obtained using the

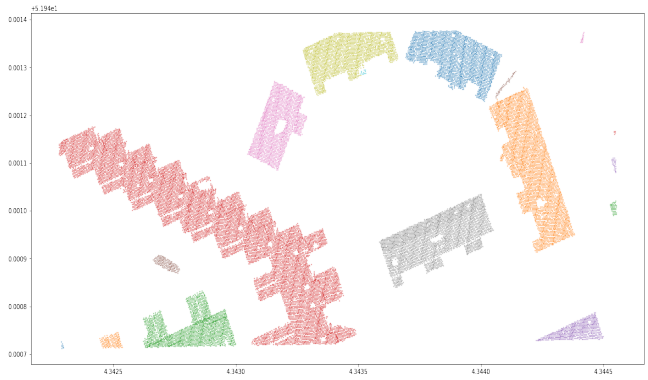

Figure 16: Clustering obtained using DBSCAN algorithm.

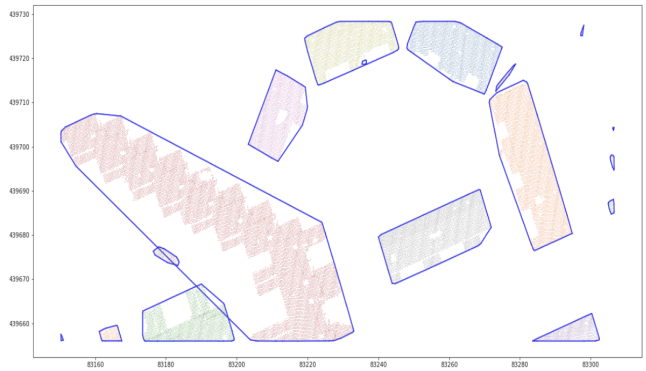

Figure 17: Footprints obtained using Convex Hull algorithm.

Alpha Shape are shown in Figure 18. We can observe that the Alpha Shape with $\alpha = 2$ provides a good approximation for the given cluster of points and can handle the non-convex nature of the buildings.

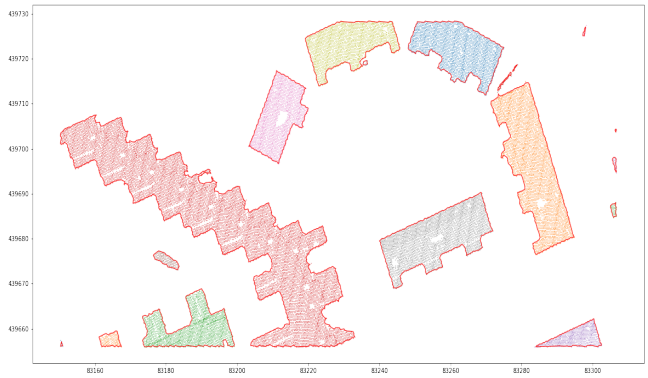

Figure 18: Footprints obtained using Alpha Shape algorithm.

4. Evaluation Metrics

Building footprints were evaluated manually using human intuition due to the lack of reference data. We visualized our results on Open Street Map [6] as shown in Figure 19. On visual comparison, we observed that our model was able to obtain a good approximation of the building footprints for the given area.

Figure 19: Visualization of building footprints using Open Street Map.

Some of the commonly used evaluation metrics on the availability of reference data are listed below:

- Pixel-based image classification metrics: Accuracy, Precision(Correctness), Recall(Completeness), F1-Score, IoU-Score.
- Object-based image classification metrics: Perimeter Ratio, Area Ratio.
- Count-based metrics: Commission and omission errors in the number of buildings identified.

5. Conclusion

We observed that the ANN classifier gave the best performance in finding the building structures from the given aerial LiDAR point clouds. Following that using 2D coordinate density-based clustering, DBSCAN, with $min\_pts = 4$ and $\epsilon = 0.5$, we were successful in identifying the 18 building clusters in our data. Further, the Alpha Shape helped us to obtain a good boundary approximation for the building clusters. With the help of satellite imagery, we can observe that the building clusters obtained from the DBSCAN do not separate the buildings which are too closely situated. This observation can also be made in the footprints extracted from the Alpha Shape, the boundary of buildings would be generally polygons with 4 sides but our model failed to find the internal boundaries of two closely situated buildings resulting in zig-zag boundaries. This is due to the uniform distribution of LiDAR data points over the building clusters which the separation difficult. Despite the above shortcoming, our model was able to extract a good approximation of the building footprints for the given area.

Possible further work can include using the ”Bayesian” approach for building footprint extraction, including prior knowledge like the typical area of a building floor in that area along with machine learning-based architecture. Another possibility is the use of supervised learning techniques to train a model to identify and complete the polygons given the number of expected edges or vertices. It would be interesting to see a comparison of these ideas and to see if the process can be generalized for LiDAR point clouds. Also, exploring the z coordinate

of LiDAR points can be another direction to identify a more optimal BFP.

Acknowledgement

The authors would like to thank Professor Preethi Jyothi from the IIT-B CSE department for her guidance on this project.

References

[1] D. Campbell, B. White, P. Arp, Modeling and mapping soil resistance to penetration and rutting using lidar-derived digital elevation data, Journal of Soil and Water Conservation 68 (6) (2013) 460–473.

[2] H. S. M. Lim, A. Taeihagh, Algorithmic decision-making in avs: Understanding ethical and technical concerns for smart cities, Sustainability 11 (20) (2019) 5791.

[3] M. Awrangjeb, G. Lu, Automatic building footprint extraction and regularisation from lidar point cloud data, in: 2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA), IEEE, 2014, pp. 1–8.

[4] A. Brunn, U. Weidner, Extracting buildings from digital surface models, International Archives of Photogrammetry and Remote Sensing 32 (3 SECT 4W2) (1997) 27–34.

[5] M. Awrangjeb, C. S. Fraser, Automatic segmentation of raw lidar data for extraction of building roofs, Remote Sensing 6 (5) (2014) 3716–3751.

[6] Resources:,

1. https://www.openstreetmap.org
2. https://towardsdatascience.com/t-sne-machine-learning-algorithm-a-great-tool-for-dimensionality-reduction-in-python-ec01552f1a1e
3. https://www.displayr.com/using-t-sne-to-visualize-data-before-prediction/
4. https://download.pdok.nl/rws/
5. https://www.openstreetmap.org/
6. https://machinelearningmastery.com/

.

[7] Wikipedia, K-means algorithm.
URL `https://en.wikipedia.org/wiki/K-means_clustering`

[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Knowledge Discovery and Data Mining, 1996.

[9] Wikipedia, Dbscan.
URL `https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering`

[10] Wikipedia, Quickhull algorithm.
URL `https://en.wikipedia.org/wiki/Quickhull`

[11] Wikipedia, Alpha shape.
URL `https://en.wikipedia.org/wiki/Alpha_shape`

[12] Voronoi diagram, delaunay and alpha complexes.
URL `https://www.youtube.com/watch?v=-XCVn73p3xs`

[13] V. Janssen, Understanding coordinate reference systems, datums and transformations, International Journal of Geoinformatics 5 (4) (2009) 41–53.

[14] W. Zhou, H. Yan, Alpha shape and delaunay triangulation in studies of protein-related interactions, Briefings in bioinformatics 15 (1) (2014) 54–64.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (11) (2020) 139–144.

[16] A. H. Jahromi, M. Taheri, A non-parametric mixture of gaussian naive bayes classifiers based on local independent features, in: 2017 Artificial intelligence and signal processing conference (AISP), IEEE, 2017, pp. 209–212.

[17] C.-C. Chang, C.-J. Lin, Training v-support vector classifiers: theory and algorithms, Neural computation 13 (9) (2001) 2119–2147.

[18] M. Pal, Random forest classifier for remote sensing classification, International journal of remote sensing 26 (1) (2005) 217–222.

[19] S. R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE transactions on systems, man, and cybernetics 21 (3) (1991) 660–674.

[20] S.-C. Wang, Artificial neural network, in: Interdisciplinary computing in java programming, Springer, 2003, pp. 81–100.