mathematics

MDPI

Article

Data-Driven Diffraction Loss Estimation for Future Intelligent Transportation Systems in 6G Networks

Sambit Pattanaik [1], Agbotiname Lucky Imoize [2,3,*], Chun-Ta Li [4,*], Sharmila Anand John Francis [5], Cheng-Chi Lee [6,7,*] and Diptendu Sinha Roy [1]

[1] Department of Computer Science & Engineering, National Institute of Technology Meghalaya, Meghalaya 793003, India; p21cs006@nitm.ac.in (S.P.); diptendu.sr@nitm.ac.in (D.S.R.)
[2] Department of Electrical and Electronics Engineering, Faculty of Engineering, University of Lagos, Akoka, Lagos 100213, Nigeria
[3] Department of Electrical Engineering and Information Technology, Institute of Digital Communication, Ruhr University, 44801 Bochum, Germany
[4] Bachelor's Program of Artificial Intelligence and Information Security, Graduate Institute of Applied Science and Engineering, Fu Jen Catholic University, New Taipei City 24205, Taiwan
[5] Scientific Research Unit (SRU), Rejal Almaa Campus, King Khalid University, Abha 61421, Saudi Arabia; sfrancis@kku.edu.sa
[6] Department of Library and Information Science, Research and Development Center for Physical Education, Health, and Information Technology, Fu Jen Catholic University, New Taipei City 24205, Taiwan
[7] Department of Computer Science and Information Engineering, Asia University, Taichung City 41354, Taiwan
* Correspondence: aimoize@unilag.edu.ng (A.L.I.); 157278@mail.fju.edu.tw (C.-T.L.); cclee@mail.fju.edu.tw (C.-C.L.)

Abstract: The advancement of 6G networks is driven by the need for customer-centric communication and network control, particularly in applications such as intelligent transport systems. These applications rely on outdoor communication in extremely high-frequency (EHF) bands, including millimeter wave (mmWave) frequencies exceeding 30 GHz. However, EHF signals face challenges such as higher attenuation, diffraction, and reflective losses caused by obstacles in outdoor environments. To overcome these challenges, 6G networks must focus on system designs that enhance propagation characteristics by predicting and mitigating diffraction, reflection, and scattering losses. Strategies such as proper handovers, antenna orientation, and link adaptation techniques based on losses can optimize the propagation environment. Among the network components, aerial networks, including unmanned aerial vehicles (UAVs) and electric vertical take-off and landing aircraft (eVTOL), are particularly susceptible to diffraction losses due to surrounding buildings in urban and suburban areas. Traditional statistical models for estimating the height of tall objects like buildings or trees are insufficient for accurately calculating diffraction losses due to the dynamic nature of user mobility, resulting in increased latency unsuitable for ultra-low latency applications. To address these challenges, this paper proposes a deep learning framework that utilizes easily accessible Google Street View imagery to estimate building heights and predict diffraction losses across various locations. The framework enables real-time decision-making to improve the propagation environment based on users' locations. The proposed approach achieves high accuracy rates, with an accuracy of 39% for relative error below 2%, 83% for relative error below 4%, and 96% for both relative errors below 7% and 10%. Compared to traditional statistical methods, the proposed deep learning approach offers significant advantages in height prediction accuracy, demonstrating its efficacy in supporting the development of 6G networks. The ability to accurately estimate heights and map diffraction losses before network deployment enables proactive optimization and ensures real-time decision-making, enhancing the overall performance of 6G systems.

Keywords: millimeter wave; building height estimation; deep convolutional neural network; image segmentation; image processing; camera projection framework

MSC: 78A45

Citation: Pattanaik, S.; Imoize, A.L.; Li, C.-T.; Francis, S.A.J.; Lee, C.-C.; Roy, D.S. Data-Driven Diffraction Loss Estimation for Future Intelligent Transportation Systems in 6G Networks. *Mathematics* **2023**, *11*, 3004. https://doi.org/10.3390/math11133004

Academic Editors: Jianhua He, Yipeng Zhou, Wei Wang and Fan Wu

Received: 24 May 2023
Revised: 30 June 2023
Accepted: 3 July 2023
Published: 6 July 2023

1. Introduction

We are living in the technological era of 5G and are on the way to 6G. Very few technology giants have already implemented 5G on a broad global scale. 5G provides a highly scalable and flexible network technology for connecting everything from everywhere to everyone [1]. One such future 6G application may be a cyber-physical intelligent transportation system (ITS). Such artificial intelligence-enabled vehicle-to-vehicle communications and vehicle-to-roadside communications use extremely high-frequency bands (EHF) such as mmWave [2]. The vehicles are equipped with advanced sensing technologies such as the Internet of Things (IoT), consumer electronics, intelligent software, and so forth [3–6]. These heterogeneous technologies have different requirements such as low latency, high bandwidth, and higher capacity. The EHF comes with higher speeds but with several challenges. For example, signals over EHF can easily get absorbed by huge buildings, trees, clouds, etc. [7]. The signals over EHF suffer significant losses due to diffraction, reflection, and so on. Hence, one of the objectives of future 6G networks is to use integrated satellite, aerial, and terrestrial networks [8] to address the limitations of EHF and ITS [9]. In this research, we focus on 6G aerial network scenarios where unmanned aerial vehicles (UAV) and electric vertical take-off and landing aircraft (eVTOL) can be used to provide line of sight (LoS) to signals originating from consumer electronic devices [10]. UAV and eVTOL are termed urban aerial devices (UAD) in this paper. UADs fly around 120 m and also use EHF. However, UADs can face diffraction losses due to flying over buildings. The awareness of network surroundings for EHF can help operators decide the communication path without obstacles between a transceiver and receiver. Hence, in this research, we focus on understanding diffraction losses in outdoor environments to improve the communication path for UADs. The diffraction loss mostly happens when the UAD flies over a building and close to the rooftops. According to Fresnel diffraction theory, diffraction is dependent on the height of the top of the obstacle above the straight line joining two ends of paths and the distance between the transceiver and receiver from the top of the obstacles [11]. Hence, the height of buildings and tall obstacles is the key parameter to estimate diffraction losses. Therefore, we focus on estimating the height of the building in this paper to improve communication in case of higher diffraction losses.

Previously, a few technologies have been implemented for determining buildings' height, which have been focused on high-spectral satellite optical image data [12], Light Detection and Ranging (LiDAR) [13] data, and synthetic aperture radar (SAR) [14,15] image data. However, these optical approaches are very costly and difficult to implement in a wide range of scenarios, e.g., obtaining the height of every building in the world. Such approaches can be costly to keep updated for UADs. Lately, street-view imagery has been used for the estimation of the height of buildings [16–18], as the procedure is easy to implement. The process incorporates a good image-capturing device to capture only panorama or street-view imagery. Technology-based companies like Google and Microsoft provide rich open-source mapping for this two-dimensional imagery, such as Google Street View [19], Streetside [20] by Microsoft, and OpenStreetMap [21] (a collaborative map project). These wide platforms provide enormous opportunities because of their intense data collection for further economic and spatial enhancement. However, such applications cannot provide the correct estimation of building heights, which can be accessed by UADs before deciding on a communication path in a real-time and automatic way.

1.1. Contributions

The following contributions are made in this paper. Our system:

- Proposes an algorithm termed Deep Learning and Image Processing-based Height Estimator (DLIPHE) to estimate a building's height using publicly available, static, non-interactive Google Street View images;
- Uses semantic segmentation [22] to identify a building and obtain its height by detecting the contour of the image using advanced image processing techniques on images from Street View;

- Accurately predicts real-time height with minimal complexity prior to data transmission, enabling efficient communication path planning;
- Creates an improved end-to-end real-time system that can use online 2D image data and footprints to determine the heights of all buildings in the path of UADs.

1.2. Organization

The rest of the paper is arranged as follows. The related works in the literature are presented in Section 2. Section 3 explains the structure of DLIPHE and presents the detection of buildings and extraction of their dimensions in detail. It also explains the extraction of two-dimensional building footprint data and the application of the pinhole camera projection framework for the final estimation of the building's height. The experimental results are reported in Section 4. The present research paper's final remarks and recommendations for further study are presented in Section 5.

2. Related Work

In this section, we present related works to estimate diffraction losses and building heights. We also present the research gaps.

2.1. Diffraction Loss Estimation Methods

ITU [11] presents detailed mathematical models to estimate the diffraction for different types of surfaces. From their proposed model, we can conclude the diffraction loss depends on the height of the obstacles and the distance between two communication ends. Several studies estimate diffraction loss for different scenarios using mathematical models [23–25]. For instance, the authors of [23] proposed mathematical models to determine the diffraction loss in a rain forest environment from the ground. Similarly, Edgar et al. proposed a simple statistical model to estimate diffraction losses due to thin isolated trees in an air-to-ground communication (satellite-to-mobile and helicopter-to-mobile) [26]. The statistical and mathematical models need manual inputs such as the height of buildings and distances between nodes to understand the attenuation. Hence, statistical models cannot determine the diffraction losses in real-time and provide automatic information to UADs for adjusting their communication path. Hence, statistical methods can be applicable to UADs. Additionally, we need to study building height estimation methods that can be used by UADs directly to optimize the propagation path. We present the related works for building height estimation in the next subsections.

2.2. Method Based on High-Resolution Optical Imagery

Methods based on VHR synthetic aperture radar (SAR) imagery [14,15,27–29] mainly use elevation, layover, and shadow to analyze the respective building's height. Studies based on high-resolution satellite images [27] estimate buildings' heights using shadows cast and acquisition geometry. Methods based on aerial LiDAR [13] data construct polyhedral building roof-tops according to structure or shape. In [30], a complete, fully trainable convolutional–deconvolutional DNN framework is proposed that allows mapping from single satellite imagery to a digital surface model for urban area analysis. A multi-sensor combination of optical imagery data and LiDAR data is fed as training data for the pipeline. To overcome geo-referencing/projection errors and sensor calibration inaccuracies, a calibrating process is proposed, which improves LiDAR and optical data alignment. The model is validated on a high-resolution dataset of central Dublin. With the help of these data, reconstruction of a 3D model from single-view aerial imagery is performed. These methods come with constraints of scalability, as they are highly expensive to obtain and require huge computation resources for UADs. These methods also depend on huge data collection and require multiple lines of communication with servers where data is analyzed. Hence, real-time planning by 6G applications cannot be feasible in extreme situations in the future. Therefore, we should focus on real-time solutions such as image processing. Section 3.1 discusses several methods to estimate height based on street scene imagery.

2.3. Method Based on Street Scene Imagery

Recently, building heights have also been estimated using deep learning approaches on satellite images, and the results of these learning approaches highly depend on the quality of the training data. In this regard, street-view data are high-quality with a deeper level of detail, are efficient, easily available, and have a low cost. For instance, methods based on street-view are easier to obtain and highly scalable. Street scenes are easily available from Microsoft Streetside [20] and from Google Street View API [19], which makes building height estimation easier to scale. Hence, street-scene data are used to estimate height. Zhao et al. [18] demonstrated a method of building height estimation using building corner and roofline detection using street-view imagery. First, building rooflines and corners are extracted from Google Street View API, based on their parameters and 3D maps footprint. Afterward, building corners are used instead of corner lines, which yields better results in camera location calibration. Then, a deep neural network-based method is applied to filter and segregate the roofline and the building's corner. Finally, the height of the building is calculated via the pinhole camera model, based on the building's roofline and valid corners from the DNN model. Such deep data used in this method are not available everywhere.

Yuan et al. [31] presented a methodology for height estimation of the buildings that employs footprint and street-view imagery of the building from two-dimensional maps. Initially, mapping between the building's footprint and the street-view imagery of the building is fulfilled, with the help of the projection metadata of the camera. Afterward, via the camera projection framework, the camera location is calibrated. This process also involves the extraction of the corner lines of the building from street-view imagery, as the location of the camera can be inaccurate due to errors in Global Positioning System (GPS). The process of mapping is again applied with the updated (after calibration) camera location and roofline of the building detected with edge detection methods. Then, the height of the building is estimated with the help of certain parameters, calibrated camera location, the footprint of the building in the two-dimensional map, and the corresponding building's roofline height in the street image.

Mou et al. [32], estimated building height using a single monocular remote sensing imagery. A convolutional–deconvolutional network architecture, encompassed with residual learning, is proposed, which frames the ambiguous modeling between the building height map and monocular remote sensing image. The network is a composition of two, i.e., a convolutional and a deconvolutional, sub-networks. The first sub-network, the convolutional network, is responsible for feature extraction from the input remote sensing imagery. It transforms the extracted input image into a high-level, complex, multidimensional representation. On the other hand, a height map is produced with the deconvolutional sub-network. A skip connection method is introduced such as residual networks for preventing the loss of fine details of the estimated height maps, as the methodology can shuttle the visual information having low resolution.

Díaz et al. [16] proposed a novel methodology for estimation of the height of the building using street-scene imagery from Google. The proposed method estimated average building heights in an image. However, they used a machine learning approach for edge detection, which makes it a heavyweight algorithm for implementation at UADs.

In this research, we propose DLIPHE for building height estimation using semantic segmentation and rooftop detection, which requires lower data input compared to the above approaches. Moreover, DLIPHE eliminates the need for using machine learning for edge detection purposes, which reduces the complexity of the proposed approach and makes it lightweight for UADs. The DLIPHE model performs a series of advanced image processing operations on the semantically segmented image for extracting building contours. These methods include extraction of the biggest connected component, contour extraction, and approximation. Finally, the building's height is estimated using a well-known 3D camera projection framework. In the following sections, the proposed algorithm is discussed in further detail, and some of the obtained results are presented.

3. Proposed Methodology

This section describes the overall technique of the proposed DLIPHE and gives the system model and the diffraction model.

3.1. System Model

In this research, we take into consideration potential applications for ITS that will be furnished with a variety of communication devices focused on the consumer and will have a wide range of demands. These communication devices use mmWave for V2V or V2R communications in outside environments. We also note that mmWave can be attenuated and suffer signal losses due to dynamic obstacles in an outside environment. Hence, in our proposed method, we deploy UADs that are flying over a building, as well as vehicles and equipped consumer devices that communicate with UADs in case of non-LoS (NLoS), as shown in Figure 1. The UADs transmit data to dedicated devices. UADs have multiple input and output antennas for transmission, whereas, in general, we think of consumer electronics as having only a single receiving antenna. The UADs are deployed in urban environments where there are several buildings and fly over 120 m and more. UADs transmit data on the 28 GHz spectrum for a faster data rate. We also assume that UADs start from the initial point going to a fixed destination point and choose the shortest trajectory path. In the environment we have assumed for the network, our attention is focused on a UAD serving as a transmitter and a receiver (RX) for the purpose of analyzing the diffraction loss that results from buildings serving as an impediment between them. We find the rooftops of buildings to be flat, which allows for the possibility of a single knife-edge diffraction as depicted in Figure 2. In such a case, we use the Fresnel diffraction model from [11] to express signal losses, as follows:

$$v = h\sqrt{\frac{2}{\lambda}\left(\frac{1}{d_1}+\frac{1}{d_2}\right)} \tag{1}$$

$$J(v) = -20\log_{10}\left(\frac{\sqrt{1-C(v)-S(v)^2+[C(v)-S(v)^2]}}{2}\right) \tag{2}$$

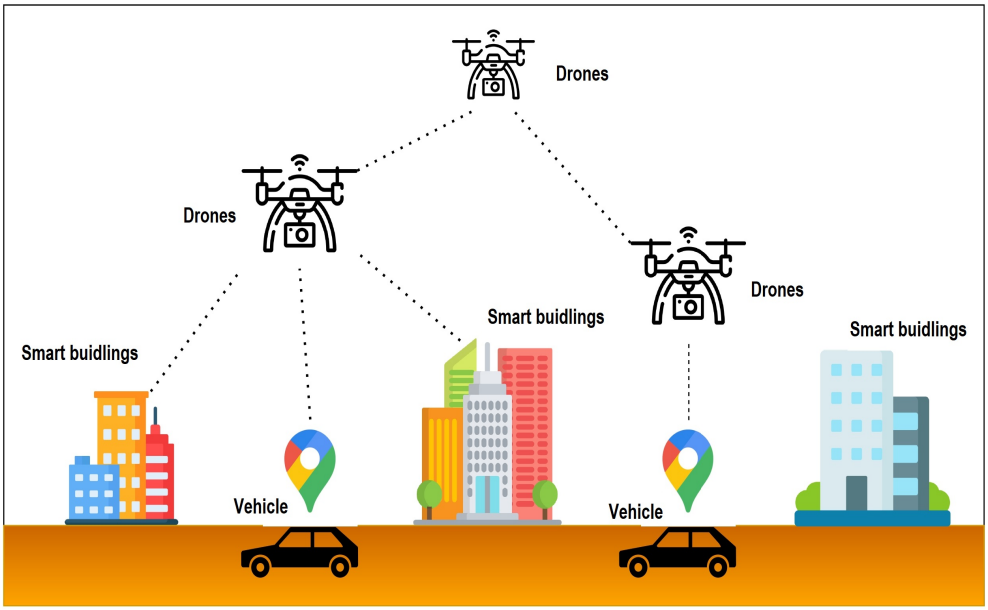


Figure 1. UAD-assisted vehicular communication over next-generation wireless network.

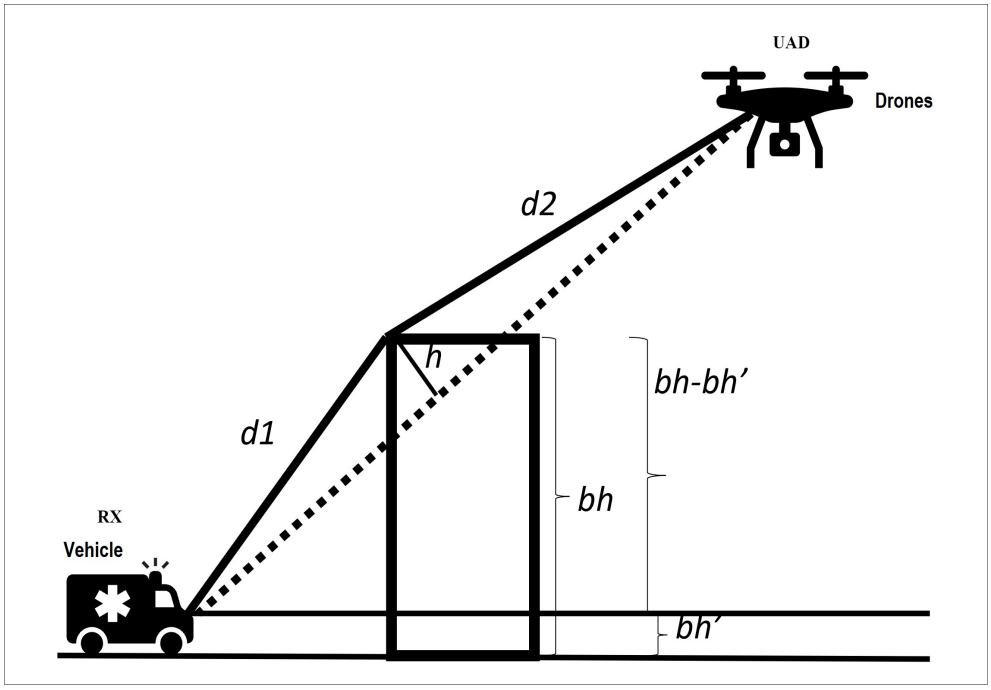


Figure 2. Diffraction model and relationship to height of building.

$C(v)$ and $S(v)$ are the real and imaginary parts of complex Fresnel integral $F(v)$, respectively. h is the height distance from the top of the building to the line joining the UAD and RX device. $d1$ and $d2$ are the distance between the top of the building and RX device, and the top of the building and UAD, respectively. λ is the wavelength. The ray diagram of this model is presented in Figure 2, where we can see that h, $d1$, and $d2$ depend on the height of the building (bh). The location of RX is available to UADs. Now, our next approach is to estimate the building heights around UADs and RX devices. Therefore, in the following section, we introduce DLIPHE as a method for Unmanned Aerial Devices (UADs) to autonomously estimate the height of buildings.

3.2. DLIPHE

Figure 3 demonstrates the process of the DLIPHE model. Initially, with the help of Google Street View API, images of the building of interest are downloaded by UADs before reaching the next destination point for proactive communication path planning. The imagery is purely static and non-interactive. Along with the API request of imagery, metadata of the same imagery can be called. Metadata contain the location of the camera, which is used in the pinhole camera projection framework. After the image of interest is obtained, semantic segmentation is applied to the image to identify a building using transfer learning. After the segmentation is completed, with the help of advanced image processing techniques, the buildings' heights are extracted. Afterward, building footprint data are extracted by calling OSM API using a Hypertext Transfer Protocol (HTTP) request. Then, after obtaining all the necessary data, computations are performed to obtain the estimated height. As illustrated in Figure 3, DLIPHE contains several steps. With the assistance of a deep learning mechanism for image processing, Algorithm 1 provides detailed instructions for estimating height.

In the following subsection, we provide a more in-depth explanation of the stages that were just discussed.

3.3. Google Street View API Call Using REST

Google provides a set of application programming interfaces (APIs) for various applications through representational state transfer (REST). We are using Google Street View API, which provides a non-interactive thumbnail of a street view or panorama via conventional

HTTP API request. To obtain the required building's image in a specific manner, we have to provide a few parameters (e.g., location, size, heading, and pitch) along with the API request. Figure 4, is an example of Google Street-View API, using HTTP request with the Python client. Both Figure 4a,b are two instances of the same building. One is taken from a closer distance with high elevation by increasing the pitch angle, whereas the other is taken from a farther distance, along the angle of the road, to capture the building's image. In urban areas, the building's imagery has to be captured from a closer distance because of the density of the buildings. In that case, the building might look a bit slanted, owing to the high elevation.

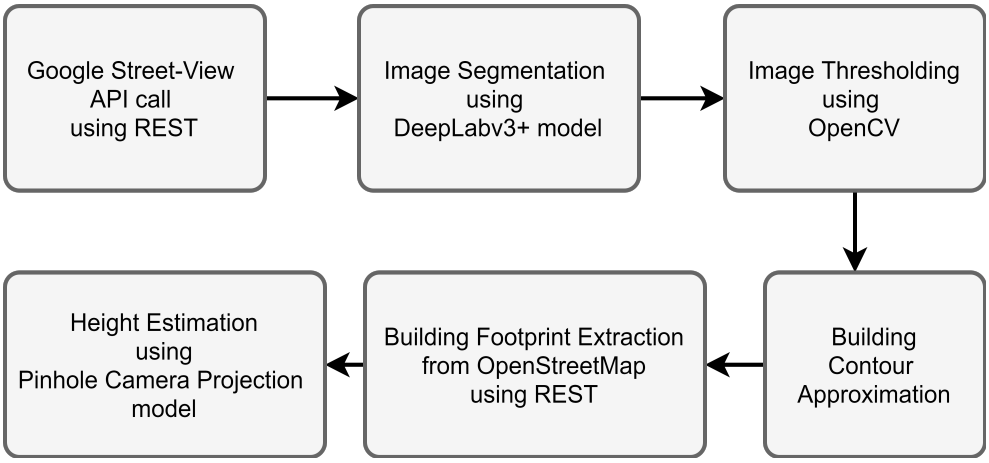


Figure 3. Steps for estimating building heights by UADs.

Algorithm 1 Deep Learning Image & Processing Based Height Estimator

Input: Google Street View image
Output: The building's image along with its estimated height
1: Request for building image from Google Street View API using HTTP request
2: Apply image segmentation with DeepLabV3+ model
3: Use the image thresholding method to extract the area of interest
4: Contour detection for extracting the topmost pixel of the building in the image
5: Building footprint extraction
6: Building height estimation using the pinhole camera projection framework
7: **return** Building image with an estimated height

(**a**) Near view (**b**) Far view

Figure 4. Image obtained from Google Street View.

The Street View API also allows for downloading of a metadata file containing the location (latitude/longitude) of the capturing camera. This camera location is further used to obtain the distance between the building and the camera.

3.4. Image Segmentation Using the DeepLabv3+ Model

To evaluate the building's height, the measurement of the building in the imagery needs to be extracted. Therefore, extracting the building is the first step toward obtaining the measurements. Here, we used semantic segmentation based on the DeepLabv3+ model [33].

Image segmentation is a digital image processing technique commonly used for partitioning an image into multiple segments or regions. Typically, it is used to locate the object boundaries in images. This technique is widely used in image processing, as it simplifies further analysis and makes it possible to extract certain information. Grouping pixels together is done on the basis of specific characteristics, such as intensity, color, or connectivity between pixels.

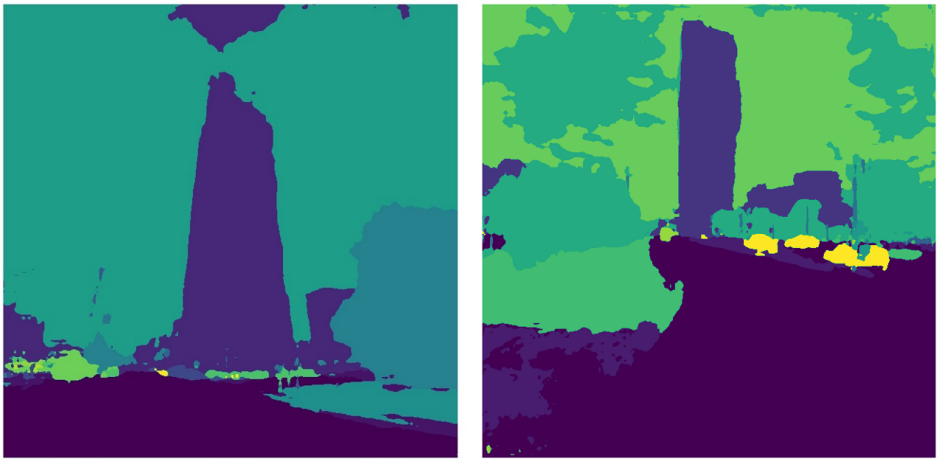
Semantic segmentation is the process of agglomerating segments of an image together that belong to a quite similar entity group. It is a pixel-level prediction, where each pixel in the imager is categorized according to a group or class. The segmentation process links each pixel to a specific class label. It is the same as the image classification at a pixel level. It is different from object detection, as it does not predict the bounding box around the objects. It does not classify between different instances of the same object. Here, for instance, in our application, we classify the pixels in the image in different class labels, such as building, road, sidewalk, sky, vegetation, etc. Common applications of semantic segmentation include self-driving vehicles, human–computer interaction, photo editing, etc.

Nowadays, deep convolutional neural networks (DCNNs) have accomplished remarkable results in numerous computer vision applications. Deep CNNs are typical feed-forward, fully-connected neural networks most often applied to analyzing imagery. Usually, DCNNs have four main operations. CNN contains the convolution operation, an activation function like ReLU or TanH, a pooling layer (max or global), fully connected layers, and an output function. The imagery is processed through a series of like operations and produces a feature vector carrying the probabilities for each class label. Many deep learning architectures have been proposed for semantic segmentation to date. Here, however, we are using Google's DeepLab model (DeepLabv3+, specifically).

The DeepLab model is a state-of-the-art deep learning model for image semantic segmentation. The DeepLab model has been implemented with a few backbone architectures, such as MobileNet [34], Xception [35], ResNet [36], etc. In this paper, we use Xception (Xception-71/XC-71 specifically) architecture as the backbone of the DeepLabv3+ model. The Xception backbone is more precise, as it has 25 times more parameters than MobileNetv2. Xception-71 is an improved version of the original Xception model. The DeepLabv3+ model is strengthened by joining a decider unit module to improve the segmentation results, especially along the entity borderline.

In this paper, the DeepLabv3+ model is used with the help of transfer learning. The model was trained on the Cityscapes dataset [37] of 50 different cities. The Cityscapes dataset mainly concerns the semantic understanding of the city's street views. The dataset has almost 30 class labels, such as road vegetation, buildings, sidewalks, etc.

The image was downloaded from Google Street View API with the help of an HTTP request and fed to the DCNN for segmentation. The output of the DCNN is a semantically segmented image with the pixels labeled as multiple classes (e.g., building, road, vegetation, etc). Figure 5a,b are the semantically segmented output of Figure 4.



(**a**) Near view (**b**) Far view

Figure 5. Semantic segmentation image.

3.5. Image Thresholding

In Figure 5, which is the output of the semantic segmentation, we can clearly see that the building has been detected in purple. There might be some sub-regions in the image, due to the presence of unwanted buildings in the background or imperfection in the DCNN.

Thereafter, thresholding is applied to the image in Figure 5 to extract the pixels that represent the building. Image thresholding is a method of segmenting or classifying pixel values in the image using a digital image processing technique. The simplest method for image thresholding is to assign a pixel as black if the intensity $I_{i,j}$ is less than some predetermined constant T (i.e., $I_{i,j} < T$). We can write this simply as:

$$if\ f(x,y) > T\ \ then\ \ f(x,y) = 0\ \ else\ \ f(x,y) = 255 \tag{3}$$

In general, the threshold value T in automatic thresholding is automatically selected by the system depending on the intensity of the object, the size of an object, the number of objects, and a fraction of an image occupied by the objects. However, automatic thresholding does not produce the expected results. For better results, we have used Otsu's thresholding method. It automatically finds an optimal threshold value based on the perceived distribution of pixels. It chooses the threshold value by minimizing the within-class variance ($\sigma_\omega^2(T)$) of the two groups of pixels separated by the thresholding operator. The implementation of Otsu's method [38] on the bi-modal image is given in Equation (4).

Consider that we have an image with L gray levels and normalized histograms, where $P(i)$ depicts the normalized frequency of i. It finds the value of T, which minimizes the gap within class variances, which lies in between the two peaks of variances of both classes. The weighted within-class variance will be given by:

$$\sigma_\omega^2(T) = q_b(T)\sigma_a^2(T) + q_0(T)\sigma_b^2(T) \tag{4}$$

where

$$q_b(T) = \sum_{i=1}^{T} P(i)\,, \qquad q_o(T) = \sum_{i=T+1}^{L} P(i) \tag{5}$$

The mean gray level value of the background and the object pixels will be:

$$\mu_b(T) = \sum_{i=1}^{T} \frac{iP(i)}{q_b(T)}\,, \quad \mu_0(T) = \sum_{i=T+1}^{L} \frac{iP(i)}{q_o(T)} \tag{6}$$

The variance of the background and the object pixels will be:

$$\sigma_b^2(T) = \sum_{i=1}^{T}[i - \mu_b(T)]^2 \frac{P(i)}{q_b(T)},\ \sigma_o^2(T) \quad = \sum_{i=T+1}^{L}[i - \mu_o(T)]^2 \frac{P(i)}{q_o(T)} \tag{7}$$

The following relations can be easily derived from the above:

$$\mu_T = \sum_{i=1}^{L} iP(i) \tag{8}$$

$$\mu_T = q_b\mu_b + q_o\mu_o\ ,\quad q_b + q_o = 1 \tag{9}$$

The above operation can be implemented using Python. The segmented image is passed to the threshold function as an input. The threshold function uses the global threshold in addition to Otsu's threshold. The threshold function results in a binary image with white (building pixels) and black (remaining background) pixels. This binary image still might contain the imperfection mentioned above. This imperfection can be removed by eliminating irrelevant sub-regions. The thresholding function can be manipulated with the help of the OpenCV library and Python, which might result in the removal of small sub-regions. Figure 6a,b are the output of image thresholding performed on segmented image Figure 5.

(**a**) Near view (**b**) Far view

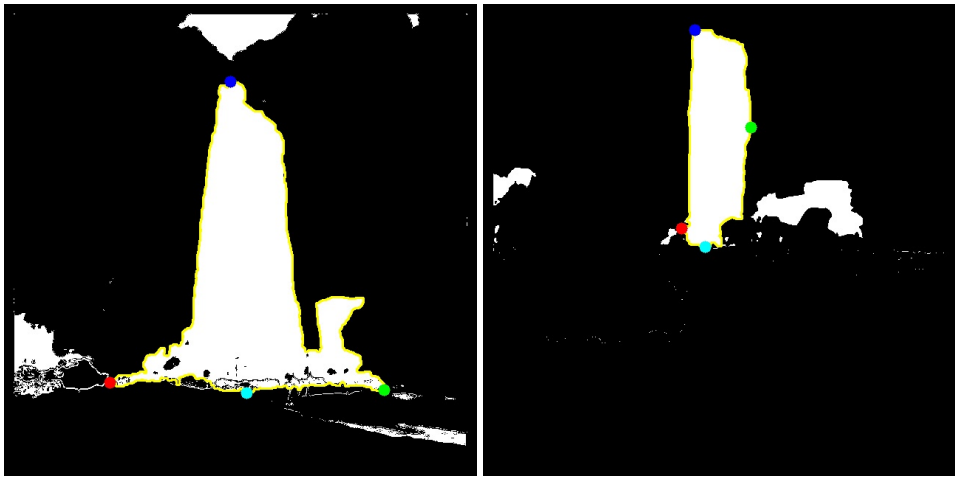
Figure 6. Image thresholding.

3.6. Contour Detection

After image thresholding, as the required building (in white pixels) is clearly visible, the contours of the foreground of the building image need to be extracted. A closed curve made by combining every continuous point containing the same color pixels or intensity is called a contour. Using contours, we can localize the objects easily.

For extracting the contour of an object in an image, the very first step could be image thresholding or edge detection, as they provide better accuracy compared to an RGB image. Image thresholding converts the RGB image into a binary image using the OpenCV library. As we already performed image thresholding in the above step, the contour approximation method is applied to the binary image.

The ultimate goal of contour detection is to extract the rooftop/topmost point or pixel of the building roofline in the image. An approximated contour is generated which contains all the quad corners (top, bottom, leftmost, rightmost). Figure 7a,b illustrate the approximated contours of the building in the image where blue, cyan, red, and green point out the top, bottom, leftmost and rightmost border, respectively.



(**a**) Near view (**b**) Far view

Figure 7. Building image with contours.

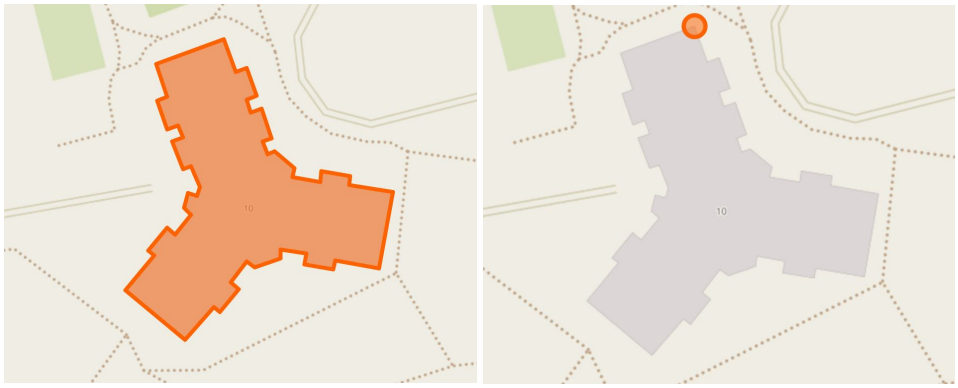
3.7. Building Footprint Extraction

Afterward, building footprint data need to be extracted for the estimation of the height of the building. The building footprint contains geo-location (latitude/longitude), structure, etc. The building's geo-location is required to identify the distance between the building and the camera. This is needed in the pinhole camera projection framework to identify the actual height. To extract information about building locations, OpenStreetMap (OSM), an open-source application, is used. OSM is a collaborative, free, editable map of the earth. It provides geographical data on the Earth. It does not have any legal or technical restrictions on its use. It has many use cases, such as geo-coding of place names and addresses, the generation of maps, and route planning. OSM is also used to obtain the building's footprint, which includes various spatial data for the building of interest.

Nominatim is a tool used to generate synthetic addresses of OSM points (reverse geo-coding) and to search ODM (OpenDroneMap) data by name and address (geo-coding). Reverse geo-coding is the method that converts a geographical location described as geographic coordinates (latitude/longitude) to human-readable addresses or place names. Geo-coding or address geo-coding is the method of converting a text-based representation of a location as the name or address of the place to the geographical coordinates (latitude/longitude). Using Python as a client, we can access the Nominatim API via HTTP request. We can use any technique, whether geo-coding or reverse geo-coding. Here, we are using geo-coding, as we are passing the address of the building. Nominatim returns an object that contains latitude, longitude, place ID, OSM type (node, ways, or relation), OSM ID, bounding box, etc. OSM ID can be used to obtain the building footprint using OSMPythonTools API via a Python client. Nodes are the points on the map (in latitude/longitude) shown in Figure 8a. A way is an ordered list of nodes, which could correspond to a street or the outline of a house. A relation is also an ordered list containing either nodes, ways, or even other relations. The latitude and longitude returned by Nominatim are for the center of the building. A list of geographical coordinates is also returned as a bounding box containing the coordinates of each node. By joining those nodes, a polygon of nodes can be built as in Figure 8b. After the nodes are obtained, we can calculate the distance between the camera (obtained from metadata of Google Street View) and the nearest node of the building.

3.8. Pinhole Camera Projection Framework

The camera projection framework explains the mathematical correlation between the geographical coordinate of a point in 3D object space and its projection onto the 2D plane imagery with the help of an ideal pinhole camera, where no single lens is used to focus light. The center of the camera is the camera aperture, described as a point or pinhole.



(**a**) Building footprint (BF) (**b**) Node BF

Figure 8. Building footprint from OpenStreetMap.

The simplest construction of the pinhole camera is shown in Figure 9. In this construction, the image plane is known as the retinal or focal plane. The pinhole O or the center of the camera is the camera aperture. The line passing through the camera center and perpendicular to the projected image plane is the principal axis.

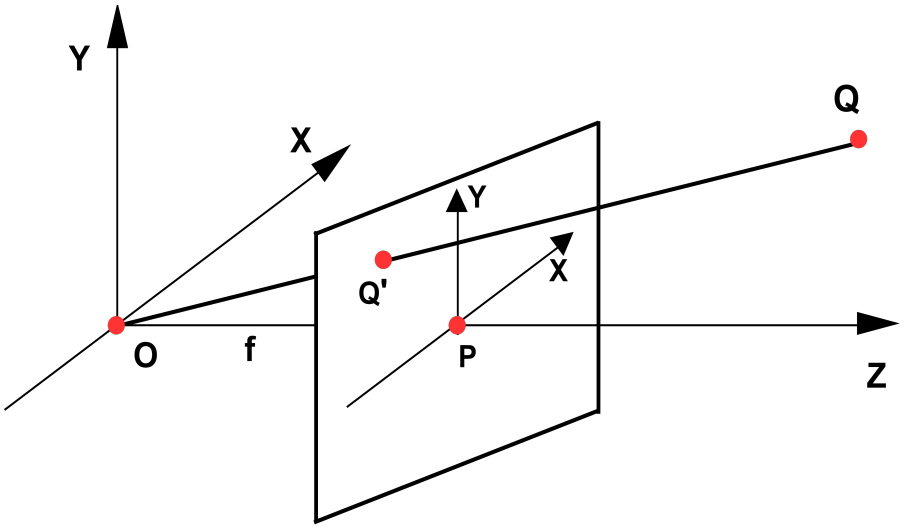


Figure 9. Camera projection 3D view.

The distance between the pinhole O and the image plane is the focal length f. The coordinate system is taken such that the origin O is in the center of the pinhole. Here, $[X\ Y\ Z]$ is the defined coordinate system centered at pinhole O. The Z-axis is perpendicular to the image. Let $Q = [X\ Y\ Z]^T$ be a point in 3D space clearly visible from the pinhole camera. After the projection onto the image plane, resulting point Q will be $Q' = [x'\ y']$, and the plane will be $Z = -f$, which is the image plane or the focal plane.

In Figure 10a, we have similar triangles, both having projection line OQ as their hypotenuses. Therefore, using the rule of similar triangles, we can derive that:

$$-\frac{x'}{f} = \frac{X}{Z}\ or\ x' = -f\frac{X}{Z} \tag{10}$$

Similarly, for the y' coordinate in the camera coordinate framework,

$$-\frac{y'}{f} = \frac{Y}{Z}\ or\ y' = -f\frac{Y}{Z} \tag{11}$$

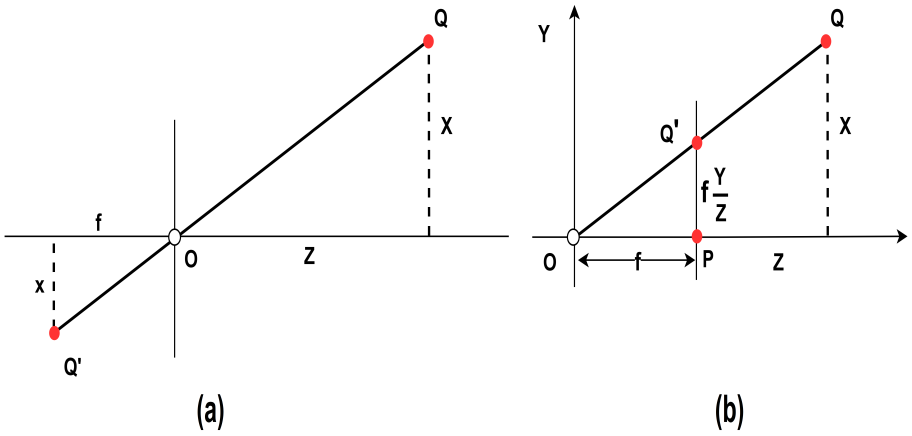


Figure 10. (**a**) The physical model of the pinhole camera. (**b**) The virtual model where the projection plane is put in front of the pinhole.

The above equation can be summarized as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = -\frac{f}{Z}\begin{pmatrix} X \\ Y \end{pmatrix} \tag{12}$$

The negative sign indicates that the image projected on the left of the pinhole camera is inverted. To eliminate the negative sign, we can mirror the image by using the virtual pinhole camera in which the plane is $Z = f$, i.e., in the same direction of the pinhole camera. Figure 10b illustrates the virtual model where the projection plane is put in front of the pinhole. Therefore, the projected point Q' of the building on the 2D image will be

$$Q' = (x',\, y')^T = (f\tfrac{X}{Z},\, f\tfrac{Y}{Z})^T \tag{13}$$

Thus, from a geometric point of view, regarding (12), f is the focal length of the pinhole, and Z is the distance between the pinhole and the building or object. X and Y are the coordinates taken as Q on the building. The central projection mapping of 3D space to 2D coordinates is

$$(X,\, Y,\, Z)^T \longrightarrow (f\tfrac{X}{Z},\, f\tfrac{Y}{Z})^T \tag{14}$$

We use Figure 11 to show the notion behind the pinhole camera projection framework and the related symbols. In the given figure, we have considered two coordinate systems, i.e., the image plane and camera coordinate system.

The camera coordinate system can be represented as $\{o, x, y, z\}$, where the o origin represents the geo-location of the camera. The image-capturing device is set parallel to the ground level. The plane xy is set perpendicular to the ground and parallel to the ground level, whereas the z-axis is perpendicular to the building's axis. The image plane coordinate system is represented by $\{o', x', y'\}$, where o' is the center of the image and $x'y'$ are parallel to the xy plane (Table 1).

Table 1. Symbol associated with the pinhole framework.

Representation	Description
f	the camera's focal length
h_r	height above images' center line
h_f	height below images' center line
c_b	building's corner point
$\hat{d}$	the distance from the camera o to the corner c_b of the building

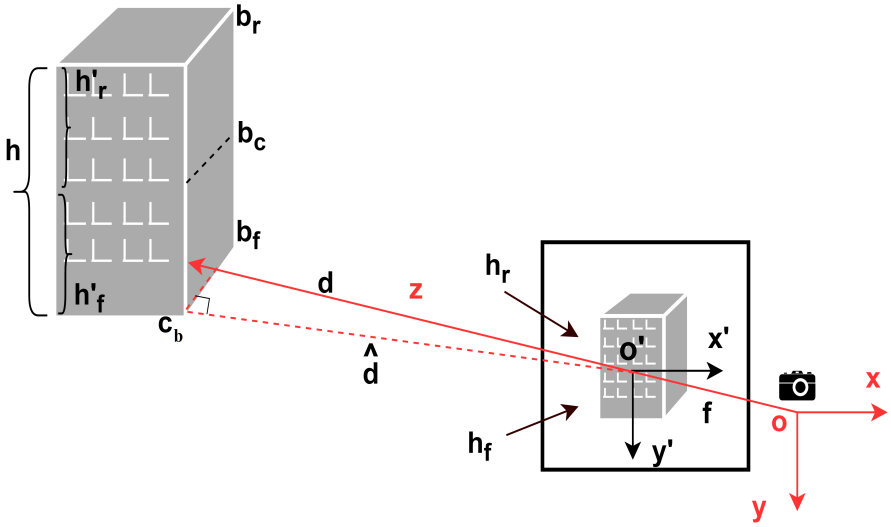


Figure 11. Pinhole camera projection framework.

Figure 11 contains a building B that has been projected onto the image. For building B, b_r, b_c, and b_f denote the roofline, the line on the building project on the x'-axis of the 2D image plane $x'y'$, and the floor, respectively. The summation of the distance from b_r to b_c, as well as the distance from b_c to b_f, is the probable height h of the building. The symbolic representation for both the distances can be taken as h'_r and h'_f. In the image plane $x'y'$, the projected length of h'_r can be written as h_r. The height of h'_f is equivalent to the height of the image-capturing car (as the camera device is mounted on the car) or human being by whom the street view is recorded, which is a constant height (here, taken as 2.5 m). Therefore, with the aforesaid information, we can derive the relation as:

$$h = h'_r + h'_f \tag{15}$$

Let us consider that $\hat{d}$ is the Euclidean distance from the image capturing device center o to the building's nearer node c_b and that f represents the focal length of the image capturing device (i.e., the euclidean distance from camera center o and the image plane center o'). The focal length f can be extracted from the image as its metadata. Therefore, the height of the building can be computed, based on the pinhole camera projection framework, as follows:

$$h = h_r * \frac{\hat{d}}{f} + h'_f \tag{16}$$

4. Experiment Methodology and Results

In this section, the evaluation of the presented system for building height estimation is given.

4.1. Dataset

In the experiment, we obtained the image dataset from Google Street View API and the building footprint (geometrical coordinates) from OpenStreetMap, respectively. For the experiments on the estimation of building height, we considered a set of buildings over 40 to 600 m. The image (640 × 640 pixels) samples have been collected from Google Street View from different cities in North America.

4.1.1. Low-Rise Buildings

The low-rise buildings dataset contains a set of small buildings with camera orientation along the street. The focal length is derived from image metadata, and the field of view is taken as 90°. The actual height of the buildings has been obtained from OpenStreetMap [21]

or Emporis (a database for building information worldwide) [39]. The camera location is given by metadata obtained from Google Street View API.

4.1.2. High-Rise Buildings

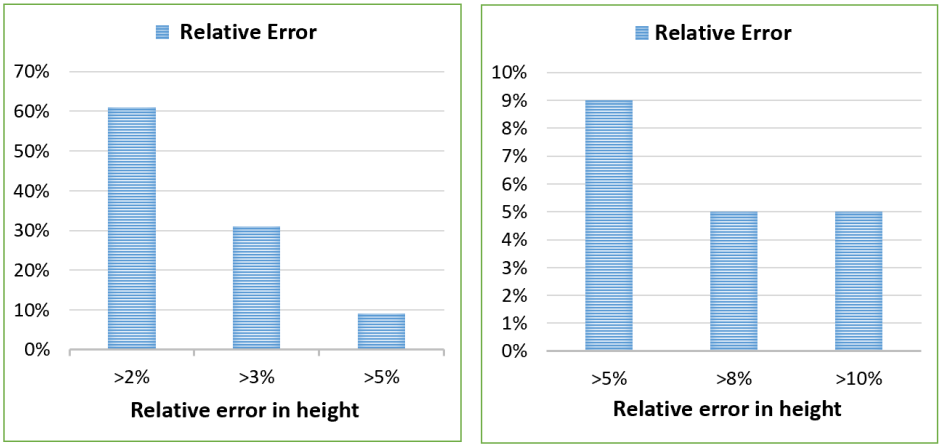
The high-rise building dataset contains a set of buildings taller than 200 m up to 600 m. The camera pitch is set up with an upward-looking view of 25° to capture the view of the building's roof. The horizontal field of view of the image is taken as 90°. The building's actual height comes from Emporis [39] or OpenStreetMap [21]. In the dataset, we considered a different set of high-rise buildings. It contains overlapped, slightly slanted, vegetation-covered, and idle-conditioned buildings.

4.2. Results of DLIPHE

The performance evaluation of the DLIPHE methodology on the low-rise and the high-rise buildings is given in this subsection. We performed these experiments on the dataset collected from Google Street View API.

4.2.1. Building Height Estimation on Low-Rise Buildings

DLIPHE was utilized to estimate the height of low-rise buildings, employing a sample set with varying heights. The calculated relative error is shown in Figure 12a. Figure 12a illustrates that the DLIPHE method accurately classified 61% of the low-rise buildings with a relative error in height of 2%. Among the remaining, 31% of the buildings had an error of 3%, and for the rest, 9% a relative error of greater than 5% was observed.



(**a**) Low-rise buildings (**b**) High-rise buildings

Figure 12. Error graph of DLIPHE on buildings.

4.2.2. Building Height Estimation on High-Rise Buildings

In high-rise buildings, the image is captured from a far distance to encapsulate the complete building's rooftop. The camera pitch angle is set upward facing, as the dataset contains buildings higher than 500 m. In urban areas, it is difficult to capture an ideal image for height estimation. Sometimes, in the case of high-rise buildings, we get slightly slanted buildings. For high-rise façades, an absolute error can be high, as the estimation of height is difficult.

As calculated for the low-rise buildings, the relative error is calculated for the skyscraper façades displayed in Figure 12b. In the graph, 9%, 5%, and 5% of buildings in the high-rise set have greater than 5%, 8%, and 10% relative error, respectively.

Table 2 shows the descriptive analysis of the experimentation performed on low-rise and high-rise buildings with their absolute estimated error and relative errors.

Table 2. Height estimation for high-rise buildings.

Absolute Error	Percentage	Relative Error	Percentage
>2 m	69.5%	>2%	60.8%
>5 m	17.4%	>5%	8.7%
>10 m	4.35%	>10%	4.35%

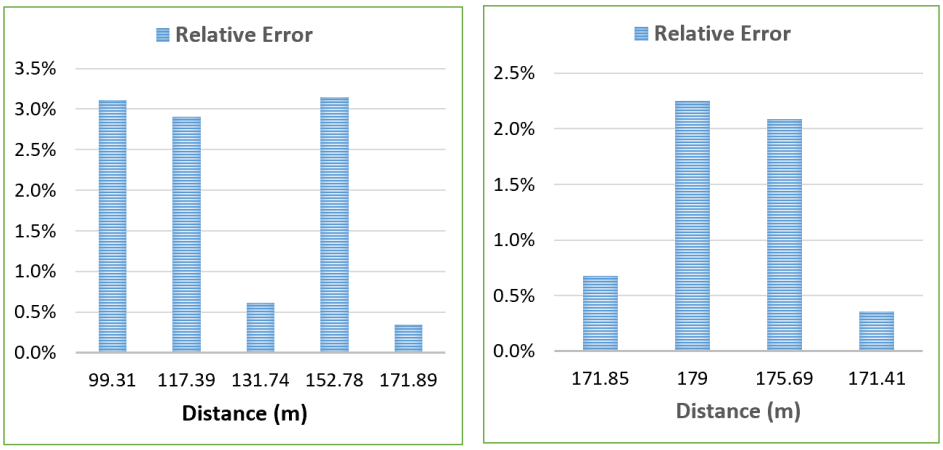
In high-rise buildings, the error may seem large due to the camera pitch movement for capturing the rooftop. However, we would like to highlight that the relative error is still admissible; e.g., since the high-rise buildings are taller than 500 m, 15 m of error in estimation is almost 3% of relative error, which is not that significant in reality.

4.2.3. Height Estimation Analysis of a Building with Variable Distances

For this experiment, we collected samples of random buildings with a height of 148 m from cities in North America using Street View. The samples were taken from different distant locations in the same direction by varying the camera–building distance. In Figure 13a, on the horizontal axis (x-axis), incremental distances from the targeted building and the capturing camera are considered in meters, whereas on the y-axis, the relative error is plotted. The graphs show that the relative error in the estimated height from varying distances is between 0.3 to 3.15%. The average relative error is equivalent to 2.0%, which is barely notable for a high-rise building of 148 m.

4.2.4. Height Estimation Analysis of a Building with Variable Directions

Sometimes, the direction from which we are capturing the building image does not provide a clear view of the targeted building because of the density of the buildings in urban areas or due to vegetation. In this situation, we would have to capture the building image from different directions that are available. In the experiment, we collected a sample of buildings from various directions, maintaining almost the same distance in each case. In Figure 13b, on the horizontal axis (x-axis), distance from the target building is taken from all possible directions in meters, whereas on the y-axis, the relative error is plotted. The graphs show that the relative error in estimated height by varying directions is between 0.3 to 2.25%, which is hardly notable for a 148 m high building.



(**a**) (**b**)

Figure 13. Relative error analysis of a low-rise building and a high-rise building with DLIPHE. (**a**) Multi-angled equidistant image capturing for building height estimation (low-rise). (**b**) Multi-angled equidistant image capturing for building height estimation (high-rise).

4.2.5. Accuracy

The estimation of the accuracy for this kind of data type would be different, as it is not a classification problem. Therefore, for calculating the accuracy of our methodology, some

measures have been considered. In this computation, the buildings that fall under <2% relative error are categorized as accurate, while the rest are errors, giving us an accuracy of 39%. We have calculated the same for <4%, <7%, and <10% of relative error, which gave us the accuracies of 83%, 96%, and 96%, respectively.

Here, for an average case, we can consider the range of <4 to <10% relative error; however, their respective accuracy lies between 83 to 96%.

4.2.6. Efficiency Evaluation Using Normalized Error

One more measure considered for the evaluation of the results obtained from DLIPHE is the normalized error (E_n). Normalized error is a statistical evaluation basically used to compare proficiency and performance measures of the testing results. It determines the confirmation of the testing. The normalized error can be evaluated as:

$$E_n = \frac{x_{ob} - x_{ref}}{\sqrt{U_{ob}^2 + U_{ref}^2}} \tag{17}$$

where (Table 3)

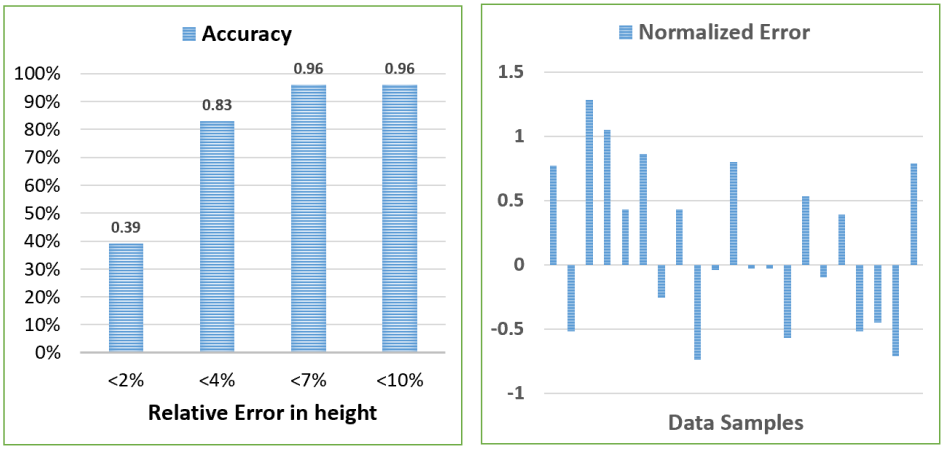
$$U_{ref} = K * U_c \;\; or \;\; U_{ref} = 1.96 * U_c \tag{18}$$

Table 3. Symbol table.

Representation	Description
x_{ob}	observed or calculated value
x_{ref}	reference or actual value
U_{ob}	expanded uncertainty of observed value
U_{ref}	expanded uncertainty of reference value
U_c	combined uncertainty (2% considered)
K	coverage factor (95%)

Using (17), the normalized error is obtained. If the value results between -1 and $+1$, then the result is acceptable; otherwise the measurement needs to be reconsidered.

Figure 14b represents the accuracy graph from the sample dataset. We randomly considered a set of estimated results and plotted the normalized error graph in Figure 14b. In the dataset, the majority of the building samples are estimated correctly, as they are in the range from -1 to $+1$. A few of the samples in the set exceed the ranges, which shows that some of the samples in the dataset have errors higher than the range and that, therefore, the estimated result cannot be considered.



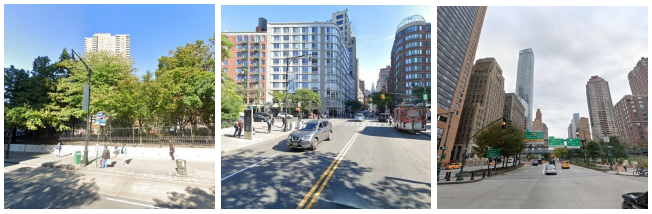
(**a**) Accuracy graph. (**b**) Normalized error.

Figure 14. Accuracy and normalized error graphs.

4.3. Analysis of Errors

In this section, a challenging scenario for the DLIPHE methodology is analyzed. Some other challenging cases will be investigated in future work.

In rural areas, buildings are often surrounded by vegetation and other objects, which makes height estimation more challenging, as it can block the entire rooftop. In this case, while performing segmentation, the algorithm will struggle to detect and classify the edges of the facade. Take Figure 15a as an example: the trees on the right-hand side are partially blocking the facade structure, which makes building identification difficult.



(**a**) (**b**) (**c**)

Figure 15. Difficult situations: (**a**) building view obscured by vegetation, (**b**) overlapping buildings, (**c**) slanted buildings.

In dense urban areas, the building view may overlap one another, and it might be difficult to detect the boundaries in a 2D image. Take Figure 15b, for example, as high-rise building boundaries are blocked by the low-rise building. Although we can try to avoid taking overlapping facade images by capturing the image from different headings or angles, sometimes it is not easy to acquire the image, especially in urban areas.

Figure 15c is an example of a slanted high-rise building. For this type of sample case, it is difficult to estimate the approximate height. This situation can also be avoided by manipulating the pitch and distance of the camera in Google Street View API. By applying the aforementioned methods, an absolute error can be slightly reduced.

5. Conclusions

Recently, there has been increasing interest in the area of EHF for future 6G networks because of unparalleled seamless and fast data delivery. However, owing to its extremely short wavelength, it suffers higher diffraction and reflection losses. UAVs and eVTOL are promising technologies to improve the propagation environment of mmWave in urban areas. In this paper, we focus on the study of diffraction losses experienced by UAVs and eVTOL, which hover over buildings. Although estimating obstacle heights in a cityscape is vital for investigating the effect on diffraction losses, most existing stochastically designed models have been studied for different scenarios and are therefore not suited for future UAM scenarios due to continuous changes in the environment by UADs. In this paper, we have proposed a building height estimation technique called DLIPHE, which uses deep learning and image processing techniques via the Street View API to capture the building's image and which then applies deep convolutional neural network-based semantic segmentation to label the building's instance. Furthermore, image processing techniques are applied for the building's contour extraction. Using the building's footprint from OpenStreetMap and the camera projection framework, the building's height is estimated. The developed technique has been tested on a set of low-rise and high-rise buildings over 50 to 600 m, respectively. Experimental results show that the proposed methodology achieves a decent accuracy on low-rise and high-rise buildings. Through various experiments, we have also tried to show that the DLIPHE achieves low relative error in high-rise buildings. The proficiency of the results achieved has also been checked with the help of normalization error. On high-rise buildings, the methodology struggles more compared to low-rise buildings. In the future, we will investigate automating the whole process, as the high-rise building needs to control the pitch angle with the respective

distance to capture the street-view image. In the future, we shall also present a method to design a communication path based on the estimation of different building heights.

Author Contributions: Methodology, S.P., A.L.I., S.A.J.F. and D.S.R.; Writing—review & editing, C.-T.L. and C.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Electronics and Information Technologies (MeitY), Government of India through grant No. 13(38)/2020-CC&BT, and Deanship of Scientific Research at King Khalid University through Small Groups Project under grant number RGP1/136/44. This work was supported in part by the National Science and Technology Council in Taiwan under contract no: NSTC 110-2410-H-165-001-MY2.

Acknowledgments: The authors express their gratitude to the Ministry of Electronics and Information Technologies (MeitY), Government of India, for their support for this research through grant No. 13(38)/2020-CC&BT. The authors also extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Small Groups Project under grant number RGP1/136/44. This work was supported in part by the National Science and Technology Council in Taiwan under contract no: NSTC 110-2410-H-165-001-MY2. The work of Agbotiname Lucky Imoize is supported in part by the Nigerian Petroleum Technology Development Fund (PTDF) and in part by the German Academic Exchange Service (DAAD) through the Nigerian-German Postgraduate Program under Grant 57473408. The authors also thank the competent bodies of OpenStreetMap, Emporis, the open-source keras library of DeepLabV3+, and Cityscapes for the resources made publicly available for researchers.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Verma, S.; Kawamoto, Y.; Kato, N. Energy-efficient group paging mechanism for QoS constrained mobile IoT devices over LTE-A pro networks under 5G. *IEEE Internet Things J.* **2019**, *6*, 9187–9199. [CrossRef]
2. Cao, H.; Garg, S.; Kaddoum, G.; Singh, S.; Hossain, M.S. Softwarized Resource Management and Allocation With Autonomous Awareness for 6G-Enabled Cooperative Intelligent Transportation Systems. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 24662–24671. [CrossRef]
3. Kim, B.; Kim, H. 6G for UAM communications: Challenges and Visions. In Proceedings of the 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 19–21 October 2022; pp. 1526–1528. [CrossRef]
4. Verma, S.; Kawamoto, Y.; Kato, N. A network-aware Internet-wide scan for security maximization of IPV6-enabled WLAN IoT devices. *IEEE Internet Things J.* **2020**, *8*, 8411–8422. [CrossRef]
5. Zeng, T.; Semiari, O.; Saad, W.; Bennis, M. Performance Analysis of Aircraft-to-Ground Communication Networks in Urban Air Mobility (UAM). In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6. [CrossRef]
6. Tiwari, P.; Lakhan, A.; Jhaveri, R.H.; Gronli, T.M. Consumer-Centric Internet of Medical Things for Cyborg Applications based on Federated Reinforcement Learning. *IEEE Trans. Consum. Electron.* **2023**. [CrossRef]
7. Zrar Ghafoor, K.; Kong, L.; Zeadally, S.; Sadiq, A.S.; Epiphaniou, G.; Hammoudeh, M.; Bashir, A.K.; Mumtaz, S. Millimeter-Wave Communication for Internet of Vehicles: Status, Challenges, and Perspectives. *IEEE Internet Things J.* **2020**, *7*, 8525–8546. [CrossRef]
8. Cui, H.; Zhang, J.; Geng, Y.; Xiao, Z.; Sun, T.; Zhang, N.; Liu, J.; Wu, Q.; Cao, X. Space-air-ground integrated network (SAGIN) for 6G: Requirements, architecture and challenges. *China Commun.* **2022**, *19*, 90–108. [CrossRef]
9. Liu, R.; Liu, A.; Qu, Z.; Xiong, N.N. An UAV-Enabled Intelligent Connected Transportation System With 6G Communications for Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 2045–2059. [CrossRef]
10. Swaminathan, N.; Reddy, S.R.P.; RajaShekara, K.; Haran, K.S. Flying Cars and eVTOLs—Technology Advancements, Powertrain Architectures, and Design. *IEEE Trans. Transp. Electrif.* **2022**, *8*, 4105–4117. [CrossRef]
11. ITU-R Recommendation. *P.526-9: Propagation by Diffraction*; ITU-R Recommendation: Geneva, Switzerland, 2005.
12. Izadi, M.; Saeedi, P. Three-dimensional polygonal building model estimation from single satellite images. *IEEE Trans. Geosci. Remote Sens.* **2011**, *50*, 2254–2272. [CrossRef]
13. Sampath, A.; Shan, J. Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds. *IEEE Trans. Geosci. Remote Sens.* **2009**, *48*, 1554–1567. [CrossRef]
14. Wang, Z.; Jiang, L.; Lin, L.; Yu, W. Building height estimation from high resolution SAR imagery via model-based geometrical structure prediction. *Prog. Electromagn. Res.* **2015**, *41*, 11–24. [CrossRef]
15. Brunner, D.; Lemoine, G.; Bruzzone, L.; Greidanus, H. Building height retrieval from VHR SAR imagery based on an iterative simulation and matching technique. *IEEE Trans. Geosci. Remote Sens.* **2009**, *48*, 1487–1504. [CrossRef]

16. Díaz, E.; Arguello, H. An algorithm to estimate building heights from Google street-view imagery using single view metrology across a representational state transfer system. In *Dimensional Optical Metrology and Inspection for Practical Applications V. International Society for Optics and Photonics*; SPIE: Baltimore, MD, USA, 2016; Volume 9868, p. 98680A.
17. Yuan, J.; Cheriyadat, A.M. Combining maps and street level images for building height and facade estimation. In Proceedings of the 2nd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics, Burlingame, CA, USA, 31 October 2016; pp. 1–8.
18. Zhao, Y.; Qi, J.; Zhang, R. Cbhe: Corner-based building height estimation for complex street scene images. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 2436–2447.
19. Anguelov, D.; Dulong, C.; Filip, D.; Frueh, C.; Lafon, S.; Lyon, R.; Ogale, A.; Vincent, L.; Weaver, J. Google Street View: Capturing the World at Street Level. *Computer* **2010**, *43*, 32–38. [CrossRef]
20. Kopf, J.; Chen, B.; Szeliski, R.; Cohen, M. Street Slide: Browsing Street Level Imagery. *ACM Trans. Graph.* **2010**, *29*, 1–8. [CrossRef]
21. Haklay, M.; Weber, P. Openstreetmap: User-generated street maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [CrossRef]
22. Chen, Y.; Weng, Q.; Tang, L.; Wang, L.; Xing, H.; Liu, Q. Developing an intelligent cloud attention network to support global urban green spaces mapping. *ISPRS J. Photogramm. Remote Sens.* **2023**, *198*, 197–209. [CrossRef]
23. Castro Eras, L.E.; Nakata da Silva, D.K.; Correia, L.; Brito Barros, F.J.; Leite de Araujo, J.P.; Protasio dos Santos Cavalcante, G. A Radio Propagation Model for a Rainforest–River Environment Using UTD and Geometrical Optics. *IEEE Antennas Wirel. Propag. Lett.* **2022**, *21*, 54–58. [CrossRef]
24. Zabihi, R.; Vaughan, R.G. Simplifying Through-Forest Propagation Modelling. *IEEE Open J. Antennas Propag.* **2020**, *1*, 104–112. [CrossRef]
25. Lee, J.H.; Choi, J.S.; Lee, J.Y.; Kim, S.C. 28 GHz Millimeter-Wave Channel Models in Urban Microcell Environment Using Three-Dimensional Ray Tracing. *IEEE Antennas Wirel. Propag. Lett.* **2018**, *17*, 426–429. [CrossRef]
26. Lemos Cid, E.; Alejos, A.V.; Garcia Sanchez, M. Signaling Through Scattered Vegetation: Empirical Loss Modeling for Low Elevation Angle Satellite Paths Obstructed by Isolated Thin Trees. *IEEE Veh. Technol. Mag.* **2016**, *11*, 22–28. [CrossRef]
27. Liasis, G.; Stavrou, S. Satellite images analysis for shadow detection and building height estimation. *ISPRS J. Photogramm. Remote Sens.* **2016**, *119*, 437–450. [CrossRef]
28. Qi, F.; Zhai, J.Z.; Dang, G. Building height estimation using Google Earth. *Energy Build.* **2016**, *118*, 123–132. [CrossRef]
29. Zeng, C.; Wang, J.; Zhan, W.; Shi, P.; Gambles, A. An elevation difference model for building height extraction from stereo-image-derived DSMs. *Int. J. Remote Sens.* **2014**, *35*, 7614–7630. [CrossRef]
30. Liu, C.J.; Krylov, V.A.; Kane, P.; Kavanagh, G.; Dahyot, R. IM2ELEVATION: Building Height Estimation from Single-View Aerial Imagery. *Remote Sens.* **2020**, *12*, 2719. [CrossRef]
31. Yuan, J.; Cheriyadat, A.M. Automatic Generation of Building Models Using 2D Maps and Street View Images. *arXiv* **2016**, arXiv:1601.07630.
32. Mou, L.; Zhu, X.X. IM2HEIGHT: Height estimation from single monocular imagery via fully residual convolutional-deconvolutional network. *arXiv* **2018**, arXiv:1802.10249.
33. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the ECCV, Munich, Germany, 8–14 September 2018.
34. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
35. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* **2016**, arXiv:1610.02357.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
37. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. *arXiv* **2016**, arXiv:1604.01685.
38. Bradski, G. The OpenCV Library. *Dr. Dobb's J. Softw. Tools* **2000**, *25*, 120–123.
39. Emporis, Provider of International Skyscraper and High-Rise Building Data! Available online: www.emporis.com (accessed on 22 May 2023).