Article

Point-Graph Neural Network Based Novel Visual Positioning System for Indoor Navigation

Tae-Won Jung ¹ , Chi-Seo Jeong ², Soon-Chul Kwon ² and Kye-Dong Jung ^{3,*}

¹ Department of Immersive Content Convergence, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea; onom@kw.ac.kr

² Department of Smart Convergence, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea; xhichi@kw.ac.kr (C.-S.J.); ksc0226@kw.ac.kr (S.-C.K.)

³ Ingenium College of Liberal Arts, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea

* Correspondence: gdchung@kw.ac.kr; Tel.: +82-2-940-5014

Abstract: Indoor localization is a basic element in location-based services (LBSs), including seamless indoor and outdoor navigation, location-based precision marketing, spatial recognition in robotics, augmented reality, and mixed reality. The popularity of LBSs in the augmented reality and mixed reality fields has increased the demand for a stable and efficient indoor positioning method. However, the problem of indoor visual localization has not been appropriately addressed, owing to the strict trade-off between accuracy and cost. Therefore, we use point cloud and RGB characteristic information for the accurate acquisition of three-dimensional indoor space. The proposed method is a novel visual positioning system (VPS) capable of determining the user's position by matching the pose information of the object estimated by the improved point-graph neural network (GNN) with the pose information label of a voxel database object addressed in predefined voxel units. We evaluated the performance of the proposed system considering a stationary object in indoor space. The results verify that high positioning accuracy and direction estimation can be efficiently achieved. Thus, spatial information of indoor space estimated using the proposed novel VPS can aid in indoor navigation.

Keywords: indoor navigation; visual positioning system; graph neural network; point cloud; three-dimensional object pose estimation; voxel

Citation: Jung, T.-W.; Jeong, C.-S.; Kwon, S.-C.; Jung, K.-D. Point-Graph Neural Network Based Novel Visual Positioning System for Indoor Navigation. *Appl. Sci.* **2021**, *11*, 9187. <https://doi.org/10.3390/app11199187>

Academic Editor: Andrea Prati

Received: 9 August 2021

Accepted: 29 September 2021

Published: 2 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Information obtained from visual positioning systems (VPSs) is more innovative than that obtained from navigation technology, such as global positioning systems (GPSs). Therefore, VPSs are increasingly being used to improve the lifestyles of people worldwide. For instance, in places where GPS service is not available, such as indoor spaces, the VPS can use the user's mobile camera to identify the surroundings and visually determine directions [1].

In this study, we propose a novel VPS method that determines the position of the camera by matching the pose information of the object estimated using the improved point-graph neural network (GNN) with the pose information of the object in a predefined voxel database. The proposed method uses a sparsely down-sampled point cloud that can be obtained from mobile devices and uses RGB information as feature information. However, unlike images, point clouds are generally sparse and are not evenly distributed on a regular grid. This implies that placing the point cloud on a regular grid generates a non-uniform number of points in the grid cells, which may lead to potential information loss in congested cells or wasted operations in empty cells when the convolution operations of same kernel size are applied to such a grid [2]. Therefore, a GNN was used to structure the point cloud, as the graph can be directly applied to the structure of the point cloud without setting it up regularly. Additionally, GNN can reuse graph edges in all layers and

need not extract the grouped and sampled points repeatedly [3]. The proposed method is a novel VPS based on point-GNN [4] that structures a three-dimensional (3D) point cloud to precisely acquire spatial information and detect objects using a GNN. The user's position can be determined by matching the pose information of the object estimated by the improved point-GNN with the pose information label of the object in the voxel database, which addresses the point cloud in the indoor space in voxel units of a certain size.

The contributions of this study can be summarized as follows:

- The proposed method estimates the pose of an object in terms of nine degrees of freedom (DOF) by improving the point-GNN.
- This is a novel method that determines the user's position as a voxel address in indoor space by matching the object pose label in the defined voxel database with the object pose estimated using the user's camera position.

Our method is a novel method of determining a position in a 3D voxel using the point cloud. The existing positioning method is an image learning-based neural network, and the general method calculates two 3D bounding boxes IoU (Intersection over Union) in seven degrees of freedom with a single rotation (around the z-axis) using a bird's eye view. Our method calculates the rotated IoU around all axes (x, y, z) with nine degrees of freedom (three transformations, three dimensions, three rotations) for pose estimation in 3D space. After calculating the 3D IoU, it is calculated with our formulas and algorithms to determine the position. We improved a new object detection approach using graph neural network on the point cloud and is a novel method of structuring point clouds by addressing 3D space with voxel labeling.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of related work. Section 3 describes the system architecture and the proposed method in detail. Experiments and performance evaluations are presented in Section 4. Section 5 discusses the inference of the obtained results, and Section 6 summarizes the study conclusions and suggests potential future applications.

2. Related Research

2.1. Point Cloud-Based 3D Object Detection

Point cloud: 3D object detection methods that use point clouds can be divided into 3D object detection based on multi-view images, voxels, and points [5]. Both multi-view image and voxel-based methods rely on the empirical quantization resolution to represent the point cloud transformation.

3D object detection based on points: The introduction of PointNet [6] and PointNet++ [7] established a method of handling irregular and unaligned point sets. This involves learning the point-by-point function directly from the original point cloud without expression transformation. A study [8] narrowed the search range in the point cloud using the frustum proposal of two-dimensional (2D) detection for the corresponding camera image and directly regressed the prediction based on the internal point. In the reported cascading frameworks [8,9], the performance of 3D object detection was severely limited by the results of 2D detection. Unlike generating proposals from images [8,9], PointRCNN [10] generated 3D proposals with high recall performance in the entire point cloud. Furthermore, PointNet++ was directly applied to perform 3D semantic segmentation of the point cloud, and 3D proposals were generated simultaneously based on each foreground point. Subsequently, these 3D proposals were used to incorporate point-by-point semantics for further improvement. To improve the directional range of the cubic anchor, a study proposed a new spherical anchor for the point cloud space [11]. Additionally, several other methods [12,13] have applied PointNet++ to point cloud processing and improved the 3D object detection performance. As point-based learning can better use each point in the original point cloud, it performs 3D object detection using the points. Point cloud primarily refers to the data collected by a light detection and ranging (LiDAR) sensor, time of flight (ToF) sensor, and RGB-D sensor. These sensors transmit light or signal to an object, record the return time, calculate the distance information for each light or signal, and generate

points to create a point cloud, which is a set of points spread in 3D space. The depth (z) information in point clouds differs from that of 2D images. Although point clouds can represent only three pieces of information, there may be additional information provided by the sensor. For instance, the LiDAR sensor can add reflectance information, whereas RGB-D can provide color information. Therefore, we can learn the 9-DoF of an object using this point cloud [14].

Point-GNN based on points: This is a GNN that detects objects in a LiDAR point cloud. Point-GNN for object detection maintains the irregularity of the point cloud based on a graphical representation rather than transforming the point cloud into a regular grid, such as an image or voxel. Unlike the technique of repeatedly sampling and grouping points, point-GNN constructs the graph only once. It uses a multilayer perceptron (MLP) to model the functions $MLP_f^t(\cdot)$; $MLP_g^t(\cdot)$ and $MLP_h^t(\cdot)$ add residual connections. All iterations use a different set of MLP^t , which is not shared between iterations. A GNN uses vertex state values after T iterations to predict both the category and bounding box of the object to which the vertex belongs. The classification branch MLP_{cls} computes multiclass probabilities, and the localization branch MLP_{loc} computes the bounding box for each class.

$$\begin{aligned}\Delta x_i^t &= MLP_h^t(s_i^t) \\ e_{ij}^t &= MLP_f^t([x_j - x_i + \Delta x_i^t, s_i^t]) \\ s_i^{t+1} &= MLP_g^t(Max(\{e_{ij}^t \mid (i, j) \in E\})) + s_i^t\end{aligned}\quad (1)$$

Subsequently, point-GNN extracts the features of the point cloud by repeatedly updating the vertex features in the graph. As point-GNN is a single-step detection method, it efficiently encodes the point cloud in an adjacent fixed-radius graph. Point-GNN is a network that predicts the category and shape of the object to which each vertex of the graph belongs. The edges of the graph connect adjacent points within a fixed radius to ensure that feature information flows between adjacent points. GNNs reuse graph edges in all layers and avoid repetitive grouping and sampling of points. Table 1 is in terms of the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) benchmark, point-GNN achieved high accuracy using only point clouds and exhibited accuracy similar to that of fusion-based algorithms [4,15–25].

Table 1. Performance comparison of the KITTI test set by submitting to the official test server.

One-Stage Method	Reference	Modality	3D Object Detection AP				Time (ms)
			Easy	Moderate	Hard	mAP	
VoxelNet [15]	CVPR 2018	LiDAR	77.82	64.17	57.51	66.5	220
ContFuse [16]	ECCV 2018	LiDAR+RGB	83.68	68.78	61.67	71.38	60
SECOND [17]	Sensors 2018	LiDAR	83.34	72.55	65.82	73.9	50
PointPillars [18]	CVPR 2019	LiDAR	82.58	74.31	68.99	75.29	23.6
TANet [19]	AAAI 2020	LiDAR	84.39	75.94	68.82	76.38	34.75
Associate-3Ddet [20]	CVPR 2020	LiDAR	85.99	77.40	70.53	77.97	60
HotSpotNet [21]	ECCV 2020	LiDAR	87.60	78.31	73.34	79.75	40
3DSSD [22]	CVPR 2020	LiDAR	88.36	79.57	74.55	80.83	38
SA-SSD [23]	CVPR 2020	LiDAR	88.75	79.79	74.16	80.90	40.1
CIA-SSD [24]	AAAI 2021	LiDAR	89.59	80.28	72.87	80.91	30.76
SE-SSD [25]	CVPR 2021	LiDAR	91.49	82.54	77.15	83.73	30.56
Point-GNN [4]	CVPR 2020	LiDAR	88.33	79.47	72.29	80.03	643

2.2. D Intersection over Union (IoU)

IoU is widely adopted as an evaluation indicator in several visual tasks, such as object detection, segmentation, and visual tracking. Additionally, it is generally computed independently of the neural network training loop without participating in the backpropagation process. To regress the parameters of the rotated bounding box, the conventional 3D object detection approach regresses the translation, scale, and yaw angle separately,

using Smooth-L1 Loss [26] based on the L1-norm of the parameter distance. As IoU does not change with the scale of the problem, a study [27] attempted to learn the IoU directly, such as predicting the IoU as a metric of non-maximum suppression (NMS) rather than calculating the IoU between the detected bounding box and ground truth bounding box. Another study [28] proposed a fitting quality network (FQNet) to directly predict 3D IoU between the sample and object in monocular 3D object data. Additionally, an algorithm for estimating the exact 3D IoU value of the bounding box was proposed [29]. The accurate 3D IoU values of 3D boxes in 3D object detection can be computed using a three-part algorithm. First, the Sutherland–Hodgman polygon clipping algorithm calculates the intersection point between the faces of two boxes. Second, the volume of intersection uses the convex hull algorithm of all truncated polygons. Third, IoU is computed based on the intersection and union volumes of the two boxes. Figure 1a illustrates the computation of the intersection points of each side by clipping the polygon to the box, and Figure 1b depicts the computation of the intersection volume using the convex hull algorithm of all intersections (green).

Figure 1. Computation of (a) intersection points is depicted in green and (b) intersection volume is indicated in green.

2.3. VPS

Camera pose estimation can be approximately divided into three categories in the existing VPSs [14,29]. The structure-based positioning method uses a local function to match the features of a query image and a point on a 3D model from 2D to 3D or an RGB-D image with a 3D model, and the camera pose is estimated. We concluded that the 3D-based method provides more accurate pose estimation owing to the precise model construction and maintenance [30]. Moreover, a model is trained from a specific image, wherein pose information serves as a learning-based localization method, and a scene can be expressed using the learned model. This learning-based positioning method predicts the match for pose estimation [31–34] or estimates camera poses, such as PoseNet [35], PoseNet2 [36], VlocNet [37], and VlocNet++ [38]. Table 2 presents the localization error of sub-meter-level accuracy 0.25 m using the image learning-based existing CNN models. In VlocNet [37], a high accuracy of 0.048 m was achieved using the Microsoft 7-Scenes dataset. The 7-Scenes dataset uses a dataset consisting of RGB-D images collected from seven different scenes in an indoor office environment.

Table 2. Comparison of localization error with existing CNN models on the 7-Scenes dataset.

CNN Models	VidLoc	HourglasPose	BranchNet	PoseNet2 [36]	Nnnet	VlocNet [37]
Localization error	0.25 m	0.23 m	0.29 m	0.23 m	0.21 m	0.048 m

3. Proposed Method

This section outlines the proposed method and describes the main modules and important algorithms in detail. This includes the object pose estimation and matching with a voxel database comprising voxel labeling and object pose estimation of an improved

point-GNN in the point cloud. Figure 2 illustrates the overall architecture of the method, which comprises two components.

Figure 2. Overview of the proposed visual positioning method (VPS). The process comprises (a) pose estimation stages of improved point-graph neural network (GNN) and (b) voxel database construction along with voxel labeling matching through VPS.

Pose estimation of GNN illustrated in Figure 2a: To structure the input point cloud in a graph, the key-point selected by the fixed-radius algorithm is used as the vertex of the graph to express the graph coordinates; RGB data are learned from a GNN. The improved point-GNN of the one-stage method estimates the classification and localization of the object box. The estimated localization based on the key module of 3D IoU can be represented as box = $(x, y, z, l, h, w, \phi, \theta, \psi)$ in the 9-DoF format.

Voxel database illustrated in Figure 2b: The indoor point cloud is addressed to voxels based on the proposed algorithm to construct the voxel database. The trained improved point-GNN can estimate the 3D pose of the object in the new point cloud and determine the user's position by matching the estimated pose information with that of the object in the voxel database.

As shown in Figure 2a, in the improved Point-GNN, the overall architecture of the improved Point-GNN includes three components: graph construction, GNN of T iterations, bounding box IoU and prediction. We use the point cloud created by the ToF camera as an input to the network. A point in the point cloud has all 3D coordinates representing properties, a state value, and a length vector. The state value is RGB. A graph is constructed by using a point as a vertex and connecting the neighbor points within the predefined and fixed radius. After the graph is constructed, the graph is re-processed using three iterations ($T = 3$) with GNN. To reduce the point cloud, we set the voxel size to 0.8 m for network training and 0.4 m for inference. For initial states of vertex, we set a fixed radius to $r_0 = 0.2$ m. $Th = 0.01$ is set in NMS. We train the proposed GNN end-to-end with batch size 1. We use stochastic gradient descent with a stair-case learning-rate decay. For each object, we use an initial learning rate of 0.01 and a decay rate of 0.1 per training step with 400 K steps. After $T = 3$ iterations of the graph neural network, the vertex state value is used to predict both the category and bounding box of the object to which the vertex belongs. The classification branch calculates multi-class probabilities. We use the mean cross-entropy loss as the classification loss. Finally, the network calculates the bounding box for each class. Add $L1$ regularization to each MLP to prevent overfitting. Since multiple vertices can be in the same entity, the neural network can output multiple bounding boxes of the same entity. We use 3D IoU and NMS to merge these bounding boxes into one. The size of MLP_{cls} is $(64, \#(class))$. The MLP_{loc} size of each class is $(64, 64, 9)$. MLP_{cls} predicts the class of an object along with the Background class and the DoNotCare class.

3.1. Point-GNN for 3D Object Pose Estimation

This section explains the pose estimation of the object box. The input of the proposed system is a point cloud created using a 3D sensor, such as ToF or LiDAR, of a mobile phone camera or other mobile platforms. We improved the existing point-GNN, which is used for detecting objects using point clouds. Moreover, it is a single-step detection method that maintains the irregularity of the point cloud using a graphical representation. A general GNN reuses graph edges in all layers, prevents repetitive grouping and sampling of points, and improves vertex features by aggregating them along the edges. In the $(t + 1)$ th iteration, each vertex function is updated in the same format as indicated in Equation (1) of Section 2.1, where e^t and v^t denote the edge and vertex features of the t th iteration, respectively. The function $f^t(\cdot)$ computes the edge feature between two vertices. Furthermore, $\rho(\cdot)$ is a set function that aggregates edge features for each vertex, and $g^t(\cdot)$ updates the vertex function using the aggregated edge function. Subsequently, the GNN outputs the vertex features or repeats the process in the next iteration.

$$\begin{aligned} v_i^{t+1} &= g^t\left(\rho\left(\left\{e_{ij}^t \mid (i, j) \in E\right\}\right), v_i^t\right) \\ e_{ij}^t &= f^t\left(v_i^t, v_j^t\right) \end{aligned} \quad (2)$$

In the case of object detection, the GNN includes the information on the object to which the vertex belongs and rearranges the state of the graph vertex. The state of the vertex is refined using the state of the neighboring vertex, and the relative coordinates of the neighboring network are used as input to $f^t(\cdot)$ for edge feature extraction. As indicated in Equation (3), relative coordinates are calculated using the key-point vertex x_j and the neighboring vertex x_i selected by the fixed-radius algorithm. The relative coordinates induce transformation invariance for the global movement of the point cloud. The point cloud of N points can be defined as a set $P = \{p_1, \dots, p_N\}$, where $p_i = (x_i, s_i)$ is a point with both 3D coordinates $x_i \in R^3$ and the state value $s_i \in R^k$ the color RGB that represents the point property.

$$s_i^{t+1} = g^t\left(\rho\left(\left\{f^t\left(x_j - x_i, s_j^t\right) \mid (i, j) \in E\right\}\right), s_i^t\right) \quad (3)$$

Object detection of point-GNN estimates the center point of the object box and the 7-DoF format of the length, height, width, and yaw values, where box = $(x, y, z, l, h, w, \theta)$. However, to estimate the pose of an object using the unfixed camera origin in a 3D indoor space, 9-DoF of the object must be estimated. Therefore, we improved two modules in the point-GNN.

First, to estimate the pose of the object, the functions of estimating the pitch and roll values were improved in addition to that estimating the yaw value. The bounding box of the object was estimated as box = $(x, y, z, l, h, w, \phi, \theta, \psi)$ in 9-DoF using the improved point-GNN. Herein, the center coordinates (x, y, z) of the object represent the center position of the bounding box; the size of the object (l, h, w) represents the box length, height, and width in the X-, Y-, and Z-axes rotations, respectively; ϕ, θ , and ψ denote the pitch, yaw, and roll angles, respectively. We encoded the bounding box with vertex coordinates (x_v, y_v, z_v) as indicated in Equation (4), where l_m, h_m, w_m, θ_0 , and θ_m represent the constant scale factors. The localization branch predicts the encoded bounding box as $\delta\text{box} = (\delta x, \delta y, \delta z, \delta l, \delta h, \delta w, \delta \phi, \delta \theta, \delta \psi)$ for each class.

$$\begin{aligned} \delta x &= \frac{x - x_v}{l_m}, \delta y = \frac{y - y_v}{h_m}, \delta z = \frac{z - z_v}{w_m} \\ \delta l &= \log\left(\frac{l}{l_m}\right), \delta h = \log\left(\frac{h}{h_m}\right), \delta w = \log\left(\frac{w}{w_m}\right) \\ \delta \phi &= \frac{\phi - \phi_0}{\phi_m}, \delta \theta = \frac{\theta - \theta_0}{\theta_m}, \delta \psi = \frac{\psi - \psi_0}{\psi_m} \end{aligned} \quad (4)$$

Second, we improved the 3D IoU for calculating the NMS value of an object in the 3D space. To estimate a single box of objects, NMS was calculated using the 3D IoU, which is a key module. Algorithm 1 uses polygon clipping to calculate the 3D IoU for the box to

determine the intersection on each side. Thus, the improved 3D IoU computes the convex hulls of all intersections and calculates the intersection volume to determine the NMS value.

Algorithm 1 3D IoU values for general 3D-oriented boxes

Input:

Box n_0 is $\text{box}_0 = (x_0, y_0, z_0, l_0, h_0, w_0, \phi_0, \theta_0, \psi_0)$

Box n_1 is $\text{box}_1 = (x_1, y_1, z_1, l_1, h_1, w_1, \phi_1, \theta_1, \psi_1)$

Applying the Sutherland–Hodgman polygon clipping algorithm:

Intersection points between faces of Box n_0 and Box n_1

Applying the convex hull algorithm:

The intersection Volume is computed using the convex hull of all clipped polygons.

Union volume is calculated as Box n_0 volume + Box n_1 volume – Intersection volume

Output: Intersection volume/Union volume

3.2. Voxel Labeling

3.2.1. Voxel Labeling in the Point Cloud Database

We divided the point cloud of indoor space into voxel units of a certain size and constructed the pose labeling of the object in the divided voxel database space. The voxel database scans the indoor space using a high-performance 3D sensor and calculates the total size of the X, Y, and Z axes of the point cloud using the maximum and minimum values of the scanned point cloud coordinates. After dividing the calculated X, Y, and Z axes of the indoor space based on the set voxel size, we performed numbering to generate the voxel address that determines the user's spatial location.

The object pose label was generated to estimate the object pose of the improved point-GNN in the built voxel database. Object pose labels comprise the voxel database object box (class, x, y, z, length, height, width, pitch, yaw, roll) and voxel number address (No.n) of the object. The center point and eight corner vertices (corner[0]–corner[7]) of the 3D box were generated based on the position of the object box in 9-DoF. Figure 3a illustrates the interior space divided by voxel size and the pose label of the object in the divided space, and Figure 3b depicts the pose label of the object box.

Figure 3. The Voxel database in the point cloud comprises (a) voxel grids and (b) corners of a box.

3.2.2. Matching the Object Pose Estimated by Deep Learning with That in the Voxel Database

The object estimated using the method presented in Section 3.1 and the object obtained in Section 3.2.1 are identical. This section explains the generation of each 3D box using the object labels obtained in Sections 3.1 and 3.2.1, match the center points of the two 3D boxes, and then match the eight vertices of the 3D box. Figure 4 illustrates the three steps, and the detailed method can be summarized as follows.

Figure 4. Translate and rotate the three-dimensional (3D) box center point of an object with voxel database coordinates.

Unification of the coordinate system illustrated in Figure 4a: The X, Y, and Z axes of the voxel database coordinate system comprise the XY plane in the ground and the Z-axis in the upward direction. The axis of most camera coordinate systems is the Z-axis in the camera optical axis direction, X-axis in the right direction, and Y-axis in the downward direction. To match the directions of the X, Y, and Z axes of the two coordinate systems, the camera coordinate system axis is rotated 90° counterclockwise based on the X-axis to unify the X, Y, and Z axes of the voxel database coordinate system.

Transformation of the center movement of the object box illustrated in Figure 4b: The Euler matrix uses Equation (5) to calculate the estimated value of the improved point-GNN object box (class, x, y, z, length, width, height, pitch, yaw, roll). Eight corner vertices, corner[0] to corner[7], of the 3D box rotated as an estimate based on the object's center coordinates are generated, and the camera origin is added to the estimated labels.

$$R_{xyz} = R_x(\phi)R_y(\theta)R_z(\psi)$$

$$= \begin{bmatrix} \cos \theta \cos \psi & -\cos \theta \sin \psi & \sin \theta \\ \cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \cos \phi \cos \psi - \sin \phi \sin \theta \sin \psi & -\sin \phi \cos \theta \\ \sin \phi \sin \psi - \cos \phi \sin \theta \cos \psi & \sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi & \cos \phi \cos \theta \end{bmatrix} \quad (5)$$

The 3D box of the object in the indoor space voxel database and that of the object estimated by the improved point-GNN are identical. The center point of the 3D_Box object (0, 0, 0) estimated based on the origin (0, 0, 0) of the center point of the 3D box in the voxel database is moved and transformed, and the center points of the 3D boxes of the two objects are matched. In both 3D boxes, the corner vertices are relative coordinates with respect to the center point.

Center rotation transformation of the object box and the camera origin rotation transformation illustrated in Figure 4c: After matching the center points of the boxes of two objects, the corner vertices of the object 3D box in the voxel database are determined to calculate the rotation angle that matches the vertices of each corner. The rotation transformation value of the camera origin in the coordinate system of the database object box is calculated based on the estimated object 3D box center point. Algorithm 2 is a pseudo code for calculating the rotation value of the camera origin based on the object's center point (0, 0, 0). One corner vertex of each of the two boxes (corners[0]) and a point on the diagonal line (corners[6]) are used to calculate the cross product and dot product of each vector.

Algorithm 2 Calculated rotation matrix between two box 3D vectors**Input:**

box1_vector is box_corners1[6]-box_corners1[0] # box_vector = [x, y, z]

box2_vector is box_corners2[6]-box_corners2[0]

Used dot and cross products to obtain the rotation matrix:

C is cross of box1_vector and box2_vector

D is dot of box1_vector and box2_vector

Norm_box1 is norm of box1_vector # used for scaling

if ~ all (C = 0) # check for collinearity

Z is [[0, -C(3), C(2)], [C(3), 0, -C(1)], [-C(2), C(1), 0]]

R is eye(3) + Z + (Z² X (1-D) / (norm(C)²))/Norm_box1² # rotation matrix

else

R is sign(D) X (norm(box2) / norm(box1)) # orientation and scaling

R is the rotation matrix, so that (except for round-off errors) ...

R * box1_vector # ... equals box2_vector

inv(R) * box2_vector # ... equals box1_vector

Therefore, if the rotation value is calculated using the rotation transformation matrix, the transformed coordinates of the camera origin rotated in the voxel database can be obtained. Equation (6) represents the rotation transformation matrix with the unit vector $u = (u_x, u_y, u_z)$ as the axis. When the camera origin is transformed into a rotation transformation matrix $R_u(\theta)$, the transformed camera coordinates in the voxel database are calculated. Consequently, the voxel, including the addressed coordinates configured in Section 3.2.1, can be determined by applying the proposed method.

$$R_u(\theta) = \begin{bmatrix} \cos \theta + u_x^2(1 - \cos \theta) & u_x u_y(1 - \cos \theta) - u_z \sin \theta & u_x u_z(1 - \cos \theta) + u_y \sin \theta \\ u_y u_x(1 - \cos \theta) + u_z \sin \theta & \cos \theta + u_y^2(1 - \cos \theta) & u_y u_z(1 - \cos \theta) - u_x \sin \theta \\ u_z u_x(1 - \cos \theta) - u_y \sin \theta & u_z u_y(1 - \cos \theta) + u_x \sin \theta & \cos \theta + u_z^2(1 - \cos \theta) \end{bmatrix} \quad (6)$$

4. Experimental Results**4.1. Voxel Labeling****4.1.1. Voxel Labeling in the Point Cloud Database**

We used deep learning datasets to select 10 types of objects for the evaluation. We considered fixed objects with a low probability of position movement in indoor space. Azure Kinect DK was used with 500 training datasets and 100 test datasets to generate point clouds for training and evaluation. Figure 5 depicts the objects selected for the self-performance evaluation.

Figure 5. Objects selected for the evaluation, including air conditioner, chair, desk, sofa_1, sofa_2, flower table, flower pot, refrigerator, round table, and podium table.

Azure Kinect DK was used to generate a point cloud for evaluating the improved point-GNN. The implementation was written in Python 3.6 using TensorFlow for graphics

processing unit (GPU) computation. Evaluation details measured the inference times on a desktop using Linux Ubuntu 16.04 LTS, Ryzen 9 3900X CPU and RTX 2080 SUPER 8G GPU (two sets). To increase the speed of estimating the object pose by reducing the number of computations in the dense point cloud generated using Azure, the improved point-GNN was learned by down-sampling 20,000 points sparse for each object in indoor space.

4.1.2. Results

Figure 6a depicts the appearance of the label of the object box during learning, and Figure 6b illustrates the graph of the key-point generated by estimating the 9-DoF based on the improved point-GNN during object box estimation.

Figure 6. (a) Learning labeling box of improved point-GNN and (b) key-point graph of object pose estimation.

Figure 7 illustrates the learning loss observed in the improved point-GNN object box estimated using the proposed method. As indicated in the figure, the proposed method verified the learning graph with a class-loss value of 88.2% of the mean average precision (mAP) of our dataset. Table 3 summarizes the results of loss values obtained by learning the 10 types of objects selected based on the improved point-GNN. We determined that high object recognition accuracy can be achieved when the learned objects are fixed to indoor external walls and floors.

Figure 7. Loss graphs obtained from the improved point-GNN learning.

The improved part of the Point-GNN used in our experiment is the code for estimating the 9DoF of the object and the 3D IoU module. Our test environment confirmed the VPS result of 732 ms in the middle-scale indoor space. To improve the speed, the IoU module we improved can be applied to a neural network with a faster operating speed.

Table 3. Loss of object pose estimation in the test dataset with the improved point-GNN.

Loss	Air Con- ditioner	Chair	Desk	Sofa_01	Sofa_02	Flower Table	Flower Pot	Refrigerator	Round Table	Podium Table
cls_loss	0.0060	0.0026	0.2075	0.0101	0.0102	0.0106	0.0077	0.0028	0.0086	0.0030
loc_loss	0.0135	0.0064	0.0102	0.0164	0.0150	0.0111	0.0244	0.0118	0.0115	0.0117
reg_loss	0.0393	0.0392	0.0393	0.0394	0.0393	0.0392	0.0394	0.0392	0.0393	0.03926
total_loss	0.0589	0.0482	0.0522	0.0597	0.0613	0.0557	0.0710	0.0539	0.0595	0.0494
recall	0.8598	0.5937	0.9825	0.9678	0.9861	0.9478	0.9760	0.3636	0.8994	0.8400
precision	0.9200	0.9047	0.9956	0.9728	0.9666	0.9695	0.9829	0.9939	0.8882	0.9767
class_mAP	0.8890	0.9092	0.9089	0.9369	0.9382	0.8538	0.8589	0.6939	0.9127	0.9185
object_loc	2.6803	0.1904	2.1270	6.2140	8.6920	1.328	2.0200	0.2542	1.6630	0.3188

4.2. VPS Results

4.2.1. Experimental Setup

In the case of point cloud data generation, the interior space of the voxel database was captured using four Leica BLK360 imaging laser scanners, which are LiDAR 3D sensors with 3D panoramic features. The point cloud comprises approximately 4.5 million pieces of RGB information with indoor space of length 35 m, height 5 m, and width 25 m. The point cloud of x, y, and z sizes were calculated using the minimum and maximum coordinates of the indoor space point cloud scanned by the LiDAR 3D sensor, and the real values of the calculated x, y, and z sizes converted into integer values increased. Although the minimum size of a voxel that can be set is 0.1 m, the smaller the size, the lower the computing speed. During the evaluation, the voxel size was set to 0.2 m. If the x, y, and z sizes are divided into voxels, the number of voxels along the X, Y, and Z axes can be calculated. The voxels can be addressed through numbering. Figure 8 depicts an interior view of the indoor space, a LiDAR 3D sensor camera, point cloud divided into voxels, and 3D doll-house view of the indoor space.

Figure 8. Leica BLK 360 imaging laser scanner scans the indoor space. The voxel database of point clouds is divided on the basis of the voxel size. This is a figure, the schemes follow the same formatting.

4.2.2. Results

Voxel information was obtained from the system to calculate the number of voxels in the X, Y, and Z axes by setting the size of the voxel based on the calculated size of the space. The point cloud of indoor information calculated the size of the indoor space in the point cloud. Furthermore, the log comprises VPS information of indoor database module. Table 4 presents the VPS log.

Table 4. Visual positioning system (VPS) log.

	Log	Details
Point cloud (Indoor)	data min _ point	[−9.5453, −1.4278, −12.1236]
	data max _ point	[13.6986, 2.6894, 21.7361]
	database min _ point	[−10, −2, −13]
	database max _ point	[14, 3, 22]
Voxel	voxel_size	0.2 m
	max_dim	[120, 25, 175]
VPS	deep Learning_object_xyz	[−0.0102, −0.0112, −0.4209]
	converted_camera_origin	[−0.0190, −0.0132, −0.4205]
	converted_VPS	[8.2809, 1.2867, 2.8794]
	voxel_points_xyz	[17.8262, 2.7146, 15.0030]
	voxels_numbers	553,696
	VPS_voxels_address	47,450

4.2.3. VPS Error Rate

The pose estimation error of the improved point-GNN is proportional to the object's center coordinates (x, y, z) and rotation values, namely the pitch, yaw, and roll. Furthermore, it is proportional to the distance of the camera's origin coordinates (0, 0, 0). However, as VPS is performed considering the voxel address, the coordinates of the camera in the database space are identical to those of the position in the cube of the voxel labeling. Figure 9 depicts the VPS error rate when the position of the camera coordinates and the actual camera coordinates are not the same voxels in the database space estimated by the VPS. On average, the error rate is as much as the distance between the centers of the two voxels.

Figure 9. Visual positioning system (VPS) error and measurement.

The actual camera position of the selected object in the indoor space can be represented as $(x_2, y_2, z_2) \in \text{Voxel No. } x_2$. The camera position predicted using VPS based on the pose estimation of the improved point-GNN can be represented as $(x_1, y_1, z_1) \in \text{Voxel No. } n$. When x_1 is the range of Equation (7) and is included in the same voxel, the proposed method provides the same position by correcting the error of the voxel space of size: length, width, height = α cm. The final actual error distance in the same voxel space does not exceed Equation (7). However, when the voxel numbers do not match, the error distance can be obtained using Equation (8).

$$(\text{VPS Error Distance} = \text{Max } \sqrt{3\alpha^2}) \in \{\text{Voxel No. } x_2 = \text{Voxel No. } x_1\} \quad (7)$$

$$(\text{VPS Error Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}) \in \{\text{Voxel No. } x_2 \neq \text{Voxel No. } x_1\} \quad (8)$$

Table 5 summarizes the VPS error distance calculated using Equation (7) in the experiment described in Section 4.1. The proposed method can localize locations within a sub-meter-level accuracy in our dataset and to test on indoor scenes we use the publicly available 7-Scenes dataset with scenes shown in Figure 10. This dataset contains significant variation in camera height and was designed for RGB-D relocalization. We compared our method with the state-of-the-art method by creating a point cloud from RGB-D images. As indicated in Tables 5 and 6, more than 80% of the test data were successfully located within 0.25 m of the ground truth location.

Table 5. Visual positioning system (VPS) error distance.

VPS Error Distance	Air Con-ditioner	Chair	Desk	Sofa_01	Sofa_02	Flower Table	Flower Pot	Refrigerator	Round Table	Podium Table
Ours	0.18 m	0.18 m	0.17 m	0.20 m	0.30 m	0.15 m	0.35 m	0.14 m	0.19 m	0.20 m

Figure 10. Localization on the 7-Scenes dataset; Chess, Fire, Heads, Office, Pumpkin, Red Kitchen and Stairs.

Table 6. Comparison of localization error on the 7-scenes dataset.

7-Scenes	Chess	Fire	Heads	Office	Pumpkin	RedKitchen	Stairs
PoseNet [35]	0.32 m	0.47 m	0.29 m	0.48 m	0.47 m	0.59 m	0.47 m
PoseNet2 [36]	0.13 m	0.27 m	0.17 m	0.19 m	0.26 m	0.23 m	0.35 m
Ours	0.25 m	0.24 m	0.19 m	0.26 m	0.28 m	0.20 m	0.32 m

5. Discussion

Figure 11 depicts the object when the point-GNN object detection learning is not performed based on the size of the object. Although the object was fixed in indoor space, with the size of the object being relatively small and the learning being performed using the improved point-GNN with a mAP of approximately 0.3, the learning was not performed with precision and recall values of 0. This is because when the graph of the improved point-GNN is generated, only a small number of key-points are included in the object. Therefore, the graph learned from the network is not generated with sufficient accuracy. This can be achieved by increasing the number of computations in the applied point cloud. However, as the down-sample and learning parameters of the point cloud were identical to the other objects used, learning of objects with dimensions $30 \times 30 \times 30$ cm or less was not performed. In addition, our method synchronized the classification learning rate and localization learning rate by setting the class loss weight and localization loss weight parameters for learning of multi-object classes for dynamic positioning system.

Figure 11. Fixed object with a small indoor space.

6. Conclusions

In this study, we proposed a novel VPS method for determining the position and pose of an object, estimated using deep learning, by matching it with the position and pose of an object in a predefined space. Initially, we estimated the position and pose of an object by learning the 3D point cloud obtained from the LiDAR camera through deep learning. Subsequently, the position and pose of the object in a predefined space were matched to determine the user's position. This confirms that the proposed system can ensure high location accuracy and direction estimation in a well-established space with structured voxels. The proposed system uses a down-sampled point cloud similar to a mobile device-based environment that can precisely acquire 3D data when GPS information is insufficient. Moreover, it enables deep learning-based visual positioning using fixed objects in an indoor environment. In summary, it is a decision system that can provide customized augmented reality content to users and also aid in indoor navigation.

Owing to the characteristics of augmented reality, which predominantly operates using mobile devices, it is necessary to investigate the work schedule of mobile GPUs to minimize the camera power consumption rate of mobile devices or the use of CPU and GPU. Thus, we intend to develop a system that complements deep learning object recognition and simultaneous localization and mapping (SLAM) for indoor navigation and augmented reality in the future.

Author Contributions: Conceptualization, T.-W.J.; Methodology, T.-W.J. and C.-S.J.; Software, T.-W.J. and C.-S.J.; Investigation, T.-W.J., C.-S.J.; and S.-C.K.; Writing—Original Draft Preparation, T.-W.J.; Writing—Review & Editing, K.-D.J. and S.-C.K.; Supervision, K.-D.J.; Project Administration, K.-D.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Code and dataset will be made available on request to the first author's email with appropriate justification. The public site for each dataset is as follows. RGB-D Dataset 7-Scenes: <https://www.microsoft.com/en-us/research/project/rgb-d-dataset-7-scenes/> (accessed on 9 August 2021).

Acknowledgments: This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2020-0-00192, AR Cloud, Anchor, Augmented Reality, Fog Computing, Mixed Reality).

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

References

1. Zhang, X.; Wang, L.; Su, Y. Visual place recognition: A survey from deep learning perspective. *Pattern Recognit.* **2021**, *113*, 107760. [CrossRef]
2. Guo, Y.; Wang, H.; Hu, G.; Liu, H.; Liu, L.; Bennamoun, M. Deep learning for 3D point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, in press. [CrossRef] [PubMed]
3. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [CrossRef] [PubMed]
4. Shi, W.; Rajkumar, R. Point-GNN: Graph neural network for 3D object detection in a point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 1711–1719.
5. Li, J.; Luo, S.; Zhu, Z.; Dai, H.; Krylov, A.S.; Ding, Y.; Shao, L. 3D IoU-net: IoU guided 3D object detector for point clouds. *arXiv* **2020**, arXiv:2004.04962v1.
6. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
7. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv* **2017**, arXiv:1706.02413v1.
8. Qi, C.R.; Liu, W.; Wu, C.; Su, H.; Guibas, L.J. Frustum PointNets for 3D object detection from RGB-D data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 918–927.

9. Wang, Z.; Jia, K. Frustum Convnet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019; pp. 1742–1749.
10. Shi, S.; Wang, X.; Li, H. PointRCNN: 3D object proposal generation and detection from point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 770–779.
11. Yang, Z.; Sun, Y.; Liu, S.; Shen, X.; Jia, J. STD: Sparse-to-dense 3D object detector for point cloud. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 1951–1960.
12. Xie, L.; Xiang, C.; Yu, Z.; Xu, G.; Yang, Z.; Cai, D.; He, X. PI-RCNN: An efficient multi-sensor 3D object detector with point-based attentive cont-conv fusion module. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12460–12467. [\[CrossRef\]](#)
13. Zarzar, J.; Giancola, S.; Ghanem, B. PointRGCN: Graph convolution networks for 3D vehicles detection refinement. *arXiv* **2019**, arXiv:1911.12236.
14. Chen, Y.; Chen, R.; Liu, M.; Xiao, A.; Wu, D.; Zhao, S. Indoor visual positioning aided by cnn-based image retrieval: Training-free, 3D modeling-free. *Sensors* **2018**, *18*, 2692. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Zhou, Y.; Tuzel, O. VoxelNet: End-to-end learning for point cloud based 3D object detection. *arXiv* **2018**, arXiv:1711.06396v12018.
16. Liang, M.; Yang, B.; Wang, S.; Urtasun, R. Deep continuous fusion for multi-sensor 3D object detection. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 641–656.
17. Yan, Y.; Mao, Y.; Li, B. SECOND: Sparsely embedded convolutional detection. *Sensors* **2018**, *18*, 3337. [\[CrossRef\]](#) [\[PubMed\]](#)
18. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, Y.; Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 12697–12705.
19. Liu, Z.; Zhao, X.; Huang, T.; Hu, R.; Zhou, Y.; Bai, X. TANet: Robust 3D object detection from point clouds with triple attention. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11677–11684. [\[CrossRef\]](#)
20. Du, L.; Ye, X.; Tan, X.; Feng, J.; Xu, Z.; Ding, E.; Wen, S. Associate-3Ddet: Perceptual-to-conceptual association for 3D point cloud object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 13329–13338.
21. Chen, Q.; Sun, L.; Wang, Z.; Jia, K.; Yuille, A. Object as hotspots: An anchor-free 3D object detection approach via firing of hotspots. In Proceedings of the European Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 68–84.
22. Yang, Z.; Sun, Y.; Liu, S.; Jia, J. 3DSSD: Point-based 3D single stage object detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 11040–11048.
23. He, C.; Zeng, H.; Huang, J.; Hua, X.S.; Zhang, L. Structure aware single-stage 3D object detection from point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 11873–11882.
24. Zheng, W.; Tang, W.; Chen, S.; Jiang, L.; Fu, C.W. CIA-SSD: Confident IoU-aware single-stage object detector from point cloud. In Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI), Online, 2–9 February 2021.
25. Zheng, W.; Tang, W.; Jiang, L.; Fu, C.W. SE-SSD: Self-ensembling single-stage object detector from point cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 19–25 June 2021; pp. 14494–14503.
26. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Jiang, B.; Luo, R.; Mao, J.; Xiao, T.; Jiang, Y. Acquisition of localization confidence for accurate object detection. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 784–799.
28. Liu, L.; Lu, J.; Xu, C.; Tian, Q.; Zhou, J. Deep fitting degree scoring network for monocular 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 1057–1066.
29. Ahmadyan, A.; Zhang, L.; Wei, J.; Ablavatski, A.; Grundmann, M. Objectron: A Large scale dataset of object-centric videos in the wild with pose annotations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 19–25 June 2021; pp. 7822–7831.
30. Sattler, T.; Torii, A.; Sivic, J.; Pollefeys, M.; Taira, H.; Okutomi, M.; Pajdla, T. Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1637–1646.
31. Brachmann, E.; Krull, A.; Michel, F.; Gumhold, S.; Shotton, J.; Rother, C. Learning 6D object pose estimation using 3D object coordinates. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 536–551.
32. Brachmann, E.; Krull, A.; Nowozin, S.; Shotton, J.; Michel, F.; Gumhold, S.; Rother, C. DSAC—Differentiable RANSAC for camera localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6684–6692.

33. Valentin, J.; Niebner, M.; Shotton, J.; Fitzgibbon, A.; Izadi, S.; Torr, P.H. Exploiting uncertainty in regression forests for accurate camera relocalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4400–4408.
34. Shotton, J.; Glocker, B.; Zach, C.; Izadi, S.; Criminisi, A.; Fitzgibbon, A. Scene coordinate regression forests for camera relocalization in RGB-D images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 2930–2937.
35. Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; Volume 31, pp. 2938–2946.
36. Kendall, A.; Cipolla, R. Geometric loss functions for camera pose regression with deep learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5974–5983.
37. Valada, A.; Radwan, N.; Burgard, W. Deep auxiliary learning for visual localization and odometry. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia, 21–25 May 2018; pp. 6939–6946.
38. Radwan, N.; Valada, A.; Burgard, W. VLocNet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4407–4414. [[CrossRef](#)]