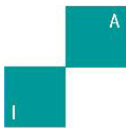
Inteligencia Artificial, 25(70) (2022), 50-63
doi: 10.4114/intartif.vol25iss70pp50-63



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Fast segmentation of point clouds using a convolutional neural network for helping visually impaired people find the closest traversable region

Paúl Tinizaray[1], José Fransisco Lucio Naranjo[1,A], Wilbert G. Aguilar[2,B,*]

[1]Escuela Politécnica Nacional, Quito-Ecuador

[2]Universidad de las Fuerzas Armadas ESPE, Sangolquí-Ecuador

[A]jose.lucio@epn.edu.ec [B]wgaguilar@espe.edu.ec

* Corresponding author

Abstract This document presents an approach to help visually impaired people find the closest-to-user traversable region in a point cloud. Our work aims to reduce the processing time of this task. The proposed approach uses a convolutional neural network to segment floor regions in the point cloud and criteria to determine which of these regions are traversable based on their size and position. We design the convolutional neural network to classify patches, allowing the floor to be searched at selected locations in the point cloud. We find that by searching only in the lower section of the point cloud, the processing time is reduced while the closest-to-user traversable region to the user is located. We evaluate our work using NYU-v2 dataset and determine that it has better balance between accuracy and processing time than similar works.

Resumen Este documento presenta un enfoque para ayudar a las personas con discapacidad visual a encontrar la región transitable más cercana al usuario en una nube de puntos. Nuestro trabajo tiene como objetivo reducir el tiempo de procesamiento de esta tarea. El enfoque propuesto usa una red neuronal convolucional para segmentar regiones del piso en la nube de puntos y criterios para determinar cuáles de estas regiones son transitables. La red neuronal convolucional está diseñada para clasificar parches, lo que permite buscar el piso en ubicaciones específicas de la nube. Encontramos que buscando solamente en la parte baja de la nube de puntos, el tiempo de procesamiento se reduce a la vez que se encuentra la región transitable más cercana al usuario. Evaluamos nuestro trabajo utilizando el conjunto de datos NYU-v2 y determinamos que tiene mejor balance entre exactitud y tiempo de procesamiento que trabajos similares.

Keywords: Machine learning, floor detection, convolutional neural network, NYU-v2 dataset.
Palabras Clave: Aprendizaje de máquina, detección de piso, red neuronal convolucional, NYU-v2.

1 Introduction

Semantic segmentation of point clouds is the task of assigning a class label to each point in a point cloud [19]. Its applications include 3D mapping, autonomous driving, mobile robotics and human assistance [7, 36, 39].

Related to human assistance, semantic segmentation of point clouds is used in the development of devices to assist visually impaired people to navigate safely in unknown environments [18, 17, 3, 9, 2]. Such devices are designed to be reliable, portable, affordable and provide real-time information [5, 31].

ISSN: 1137-3601 (print), 1988-3064 (on-line)

However, as pointed out by [18, 17], these characteristics can still be improved. In this sense, reducing the processing time of computational models helps to use more affordable and portable hardware. For example, the use of MobileNets, a lightweight model used for object detection in RGB images, allow [17] to detect objects at 14 fps using a Raspberry Pi model B. Also, by using PeleeNet, another lightweight model used for object detection, [3] can detect objects at 8 fps using a smartphone.

We propose an approach to quickly find the closest-to-user traversable region in a point cloud. Different from existing solutions that look for all floor regions, we only search those located in the lower section of the point cloud. Such regions are closer to the user than those located in the upper section. Furthermore, lower section concentrates floor regions [6]. This way, we can reduce the processing time.

An overview of our approach is presented in Figure 1. Given a point cloud, we first align it using the rotation matrix provided by the device that generated it. Then, the aligned point cloud is divided into lower and upper sections. To segment floor regions in lower section, we develop a convolutional neural network (CNN). Once floor regions have been segmented, we apply 3 criteria to determine which ones are traversable: if there are no obstacles above the region, if the region is attached to the bottom border of the point cloud and if the region is wide enough for the user to pass through. Finally, among traversable regions, we select the closest to the center of the point cloud.

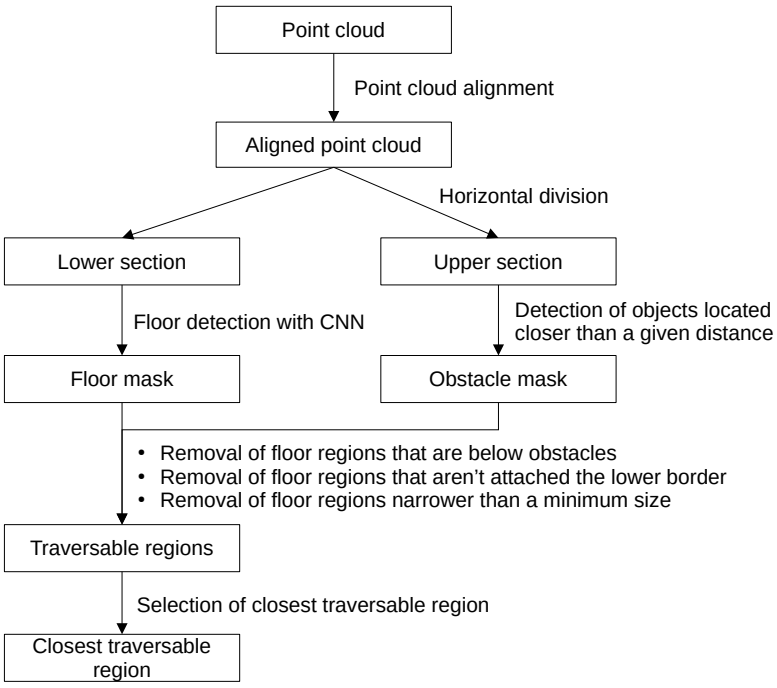


Figure 1: Approach overview.

We test our approach using NYU-v2 dataset [24] and compared it with approaches for semantic segmentation and traversable regions detection. Results show that our approach has better balance between accuracy and processing time, making it suitable to quickly find the closest-to-user traversable region in a point cloud. The main contributions of our work are:

- Development of a new CNN to segment floor regions in point clouds.
- Proposal of criteria to assess traversability of floor regions.
- Development of an algorithm to segment the closest-to-user traversable region in point clouds that is faster than related works.

The rest of the paper is structured as follow. Section 2 presents the state of the art of the art related to traversable region detection. Section 3 shows our approach for detecting the closest-to-user traversable region in a point cloud. Section 4 presents the results of the experiments conducted using our approach. Finally, Section 5 presents the conclusions and future works.

2 Related Works

To detect traversable areas in point clouds, we found 2 approaches: floor plane detection and semantic segmentation.

2.1 Floor plane detection

The goal of this approach is to identify the elements of a point cloud that belongs to floor plane. If the point cloud is defined in a cartesian coordinate system, it is possible to assume that these points are contained in a plane described by Equation 1. To find the parameters of Equation 1, [3, 27, 33, 2, 1, 26, 15] use RANSAC while [34] tackle it as a linear optimization problem. Since more than one plane may exist in the point cloud, the one that corresponds to the floor could be selected by the number of inliers [33, 1], the distance of the plane from the origin [3, 26] or the normal orientation of the plane [2, 15].

$$Ax + By + Cz + D = 0 \tag{1}$$

If it is assumed that the floor plane is parallel to the xz plane, points of interest could be found by selecting those with y-coordinate close to perpendicular distance between the device that captured the point cloud and the floor [20, 21, 38], those with normal unit vector close to $(0, 1, 0)$ [23, 37] or a combination of both [16, 13].

2.2 Semantic segmentation

To perform semantic segmentation in a point cloud, Wolf *et al.* [35] introduce a model called 3D entangled forest classifier, that is a version of the random forest classifier modified to learn which labels are likely to occur close to each other and in which specific geometric configuration.

Liu *et al.* [18] presents a version of PointGroup, modified to increase prediction capability while maintaining a small number of parameters. After segmenting the point cloud, a 2D top-view representation is generated to determine objects' localization and passable regions.

In [32], a point cloud is voxelized and processed by a 3D fully convolutional neural network to produce voxel labels. A trilinear interpolation layer transfers voxel labels back to the original point cloud representation. The obtained point scores are used for inference in the 3D fully connected CRF to produce the final output.

Schulz *et al.* [28] divide an RGB-D image into patches of size proportional to their depth values and classify them using a CNN. Such CNN is trained using 8 input features: 3 corresponding to RGB channels, 4 containing a simplified histogram of oriented depth and 1 for the height above the ground.

In [11, 22], a point cloud is first voxelized. Then, it is divided into cubic-shaped groups of voxels. Each group is classified using a 3D CNN. The inferred labels are mapped back to the original point cloud. To train the 3D CNN, [11] use the occupancy values of the voxels while [22] use the values of point density and mean elevation of each voxel.

Our approach works on patches of a point cloud because, as pointed by [22], sliding window based techniques are usually computationally cheaper.

3 Proposed approach

The proposed approach is detailed in Algorithm 1. A cartesian point cloud with grid-like structure must be provided. Such point cloud can be obtained using an RGB-D sensor or a stereoscopic sensor. To use the same coordinate system in all point clouds, the first step consists in align the point cloud using the rotation matrix provided by the device that generates it.

Next, the point cloud is divided horizontally into 2 sections. The goal of this division is to increase the speed of floor identification by performing it only in lower section.

The process of floor detection is presented in Figure 2. The lower section is divided into patches using a regular grid. Each patch is classified using a CNN to obtain a label. The label is assigned to the floor mask in the corresponding position of the patch.

If there is at least one floor region, traversable region/s are identified. For this purpose, we propose 3 criteria:

1. If there are not obstacles above the floor region, it is traversable.
2. If the floor region is attached to the bottom of the mask, it is traversable.
3. If the floor region is wide enough for the user to pass through, it is traversable.

To apply the first criterion, the upper section of the point cloud is processed by identifying points located closer than a safe distance thus obtaining the obstacle mask. Then, columns in floor mask are reset if their corresponding column in obstacle mask is occupied.

To apply the third criterion, regions narrower than a given width are reset. The actual width of regions in floor mask can be calculated by superimposed them in the point cloud.

After identifying traversable regions, if there is at least one floor region, the closest region to the center of the mask is selected as the output of the algorithm. Such region is closer to the user and is faster to reach.

Algorithm 1: Proposed approach

```
Input: Point cloud, sensor rotation matrix, division value, safe distance, floor detection CNN,
       minimum region width value
Output: Mask with closest region or empty mask
Align the point cloud using the sensor rotation matrix;
Divide point cloud horizontally using the division value;
Get floor mask by processing the lower section with floor detection CNN;
if There are any floor regions in floor mask then
    Get obstacle mask by identifying elements located closer than safe distance in the upper
     section;
    if There are any obstacle regions in obstacle mask then
        Get indices of columns occupied by obstacles in obstacle mask;
        Reset elements in floor mask that belongs to columns indicated by retrieved indices;
    if There are any floor regions in floor mask then
        Reset regions that are not attached to lower border;
        Reset regions narrower than minimum region width value;
        if There are any floor regions in floor mask then
            Select region closest to center of the mask;
            return Mask with closest region;
        else
            return Empty mask;
    else
        return Empty mask;
else
    return Empty mask;
```

3.1 Proposed CNN

Similar to [22, 11], our CNN is based on LeNet [14]. It is composed by 3 convolutional layers, 3 pooling layers and 2 fully connected layers.

CNNs are specialized neural networks for processing data with a known grid-like topology [8]. They include convolutional, pooling and fully connected layers. Convolutional layers are defined by kernel size

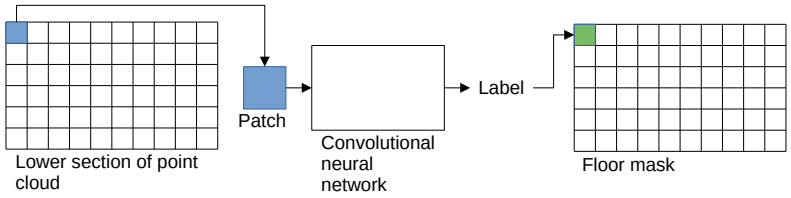


Figure 2: Floor detection process.

(K), number of channels (C), step (S), type of padding (P) and activation function (AF). Pooling layers are defined by kernel size, step, padding and the pooling operation (O). Fully connected layers are defined by the number of neurons (n) and activation function. [12] offer more information about CNN elements.

4 Experiments

4.1 Dataset

Like [35, 28], we used NYU-v2 dataset to evaluate our approach. This dataset is composed by 1449 640x480 RGB-D images, 795 for training and 654 for evaluation. For each image, a dense per-pixel labeled mask is provided. In this work, 4 classes were distinguished: ground, structure, furniture and props. Figure 3 shows the percentage of elements per class in training masks. It is noted that this dataset is unbalanced.

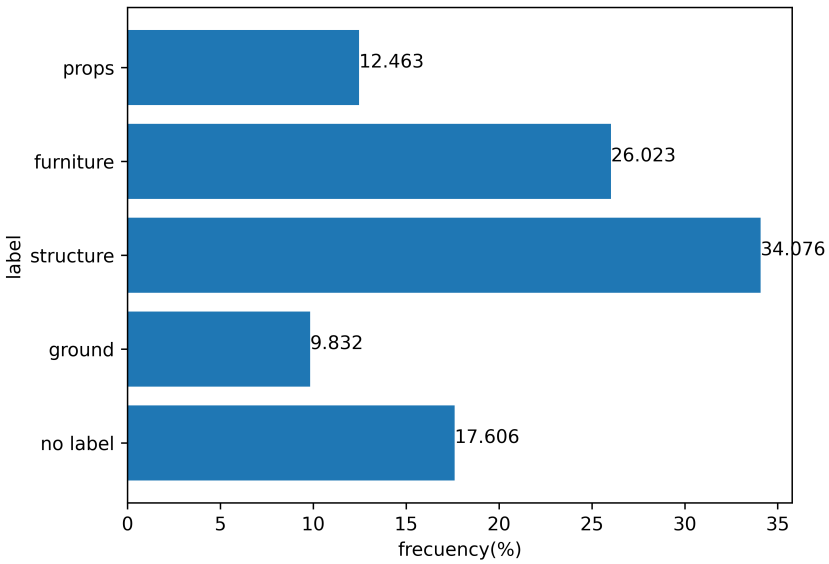


Figure 3: Percentage of elements per class in training masks.

4.2 Experiments with the CNN

4.2.1 Hyperparameters selection

For convolutional layers, hyperparameters K = 3, S = 1 y P = 1 were selected because of [30, 29, 25] recommendation. Since values of point cloud can be positive as well as negative, hyperbolic tangent

(tanh) was selected as activation function. For input size, number of filters in convolutional layers and number of neurons in the first fully connected layer, we explored the following values:

- **Input size**: (8, 8, 3), (16, 16, 3), (32, 32, 3), (64, 64, 3).
- **Number of filters in convolutional layers**: 16, 32, 64.
- **Number of neurons in first fully connected layer**: 16, 32, 64.

The different configurations of our CNN were developed using Tensorflow 2.4.1 and Keras 2.4.3 libraries. To configure the training process, we used default values of the hyperparameters, except for those indicated in Table 1. Also, since NYU-v2 is unbalanced, we provided a vector containing class weights.

Table 1: Hyperparameters used for CNN training.

Hyperparameter	**Value**
Epochs	64
Patience	6
Loss function	Cross entropy

We use 80% of the training set to train the configurations and reserved 20% to validate them. To generate examples and targets, point clouds and their corresponding masks were divided into patches. For each mask patch, if labels of all elements were the same, its corresponding point cloud patch and the unique label were stored as example and target respectively.

Similar to [35], class and balanced accuracy were used to evaluate the configurations. Results achieved by these configurations are available in Appendix A. The one with input size (16, 16, 3), 16 filters in convolutional layers and 64 neurons in first dense layer achieved higher floor class accuracy in validation set. We identified this configuration as model 16-16-64. The structure of model 16-16-64 is summarized in Table 2.

Table 2: Structure of the proposed CNN.

Layer	**Parameters**
Input	size = 16x16x3
Convolutional	K = 3, C = 16, S = 1, P = 1, AF = tanh
Pooling	K = 2, S = 2, O = average
Convolutional	K = 3, C = 16, S = 1, P = 1, AF = tanh
Pooling	K = 2, S = 2, O = average
Convolutional	K = 3, C = 16, S = 1, P = 1, AF = tanh
Fully connected	n = 64, AF = tanh
Fully connected	n = 4, AF = softmax

Using model 16-16-64, we also explored different values of batch size and learning rate as follows:

- **Batch size**: 32, 64, 128, 256, 512
- **Learning rate**: 0.001, 0.0001

Results achieved by these configurations are available in Appendix B. The one with batch size 32 and learning rate 0.001 achieved higher balanced accuracy in validation set.

4.2.2 CNN evaluation

To reduce the effect of randomness of weights initialization, model 16-16-64 was trained 10 times with different initial weights. Table 3 shows average scores achieved by the proposed CNN in NYU-v2 evaluation set along with the results of [35]. Regarding the processing time, the proposed CNN achieved 149.037 ms. per point cloud running on an i7 laptop 8 x 2.2 GHz the method proposed by [35] achieved 730 ms.

using an i7 laptop 8 x 2.4 GHz. Figure 4 present examples of predicted masks. Each row corresponds to a different scene. First column shows the corresponding image. Second column presents the ground truth. Last column shows the predictions.

Table 3: Accuracy scores and runtime for 4 semantic classes of NYU-v2 dataset. Best values in bold.

Model	**Accuracy(%)**					**Runtime(ms)**
	Ground	Struct	Furnit.	Props	Balanced	
3-D Entangled Forests [35]	**98.800**	**87.900**	**73.900**	**40.400**	**75.025**	730.000
Proposed CNN	93.301	76.489	60.264	23.312	63.345	149.037

Table 4 shows scores achieved by the proposed CNN in the detection of ground class against the other ones. We also evaluated the method proposed by [17] for ground segmentation. This method proposes looking for floor regions only in a proposed region. Limits of such region are related to floor height. In the same way, we decide to look for only in regions below -1.109 m. We selected this value because the mean floor height of training scenes is -1.309 m. Then we provide a threshold of 0.2 m. Regarding the processing time, the proposed CNN achieved 84.048 ms. per point cloud running on an i7 laptop 8 x 2.2 GHz while the method proposed by [17] achieved 103.335 ms. running on the same device.

Table 4: Accuracy scores and runtime for 2 semantic classes of NYU-v2 dataset. Best values in bold.

Model	**Accuracy(%)**			**Runtime(ms)**
	Ground	**Others**	**Balanced**	
RANSAC-based model [17]	88.916	96.289	92.578	103.335
Proposed CNN	**92.685**	**96.806**	**94.751**	**84.048**

4.3 Traversable region detection

Floor segmentation was performed in lower fifth section of points clouds. The safe distance was set at 0.8 m. because of the average stride of a human is 0.780 m. [4]. The minimum floor region width was set at 32 in. ($\sim$ 0.8128 m.) since it is the minimum width for a door threshold according to [10]. Figure 5 shows examples of closest-to-user traversable area detection. Each row corresponds to a different scene. First column shows the RGB image. Second column presents the ground truth in green. Last column shows the segmented closest-to-user traversable region in green.

To measure execution time of our approach, all 1449 samples of NYU-v2 dataset were processed. The resulting average time was 43.789 ms. running on an i7 laptop 8 x 2.2 GHz and 83.708 ms. on a Raspberry Pi 4 Model B. Note that some instructions of the algorithm depend on whether the floor mask found is empty or not. For this reason, the average time measured in a different dataset may vary.

5 Conclusions

Regarding the detection of ground class in NYU-v2 dataset with the proposed CNN, we observed a dependency between the class accuracy of scenes and their mean floor height. Figure 6 presents the mean floor height and ground class accuracy corresponding to scenes of Figure 4. A red dashed line represents the mean floor height of training scenes, which corresponds to -1.309 m. Scenes with mean floor height closer to this line have higher ground class accuracy.

Although our network could not improve the accuracy scores obtained by Wolf *et al.* [35] in NYU-v2 dataset, we were able to reduce the processing time. While the difference in accuracy between the 2 models is 5.499%, our model works 4.898 times faster. We also compared a RANSAC-based model designed for ground segmentation, which has less parameters than our CNN. This model obtained less accuracy and demanded more processing time. We conclude that our CNN has better balance between accuracy and computational cost than other available solutions.

Different from related works, our approach looks for floor regions only in the lower section of the point cloud. As shown in Figure 5, we were able to retrieve the closest-to-user traversable area. At the

Figure 4: Examples of predicted masks. Each row corresponds to a different scene. First column shows the RGB image. Second column presents the ground truth. Last column shows the predictions. Ground = green, structure = yellow, furniture = red, props = cyan.

same time, we managed to reduce the processing time of this task. For example, [17] reports 357.680 ms running on a Raspberry Pi 4 Model B while our approach achieved 83.708 ms running on the same hardware. Similarly, [9] reports 50.000 ms running on a i7 2.2 GHz. while our approach achieved 43.789 ms running on the same hardware. Also, the processing time of our approach allows it to be part of an ETA system since young adults take 387.096 ms. to perform 1 step [4]. This means that our approach can process 8 scenes on the i7 8 x 2.2 GHz. and 4 scenes on the Raspberry PI Model B before the user changes position.

In the future, we will work in improving the accuracy and proccessing of the proposed CNN. Also, we plan to extend the usability of our approach to other datasets.

References

[1] K. Al-Muteb, M. Faisal, M. Emaduddin, M. Arafah, M. Alsulaiman, M. Mekhtiche, R. Hedjar, H. Mathkoor, M. Algabri, and M.A. Bencherif. An autonomous stereovision-based navigation system

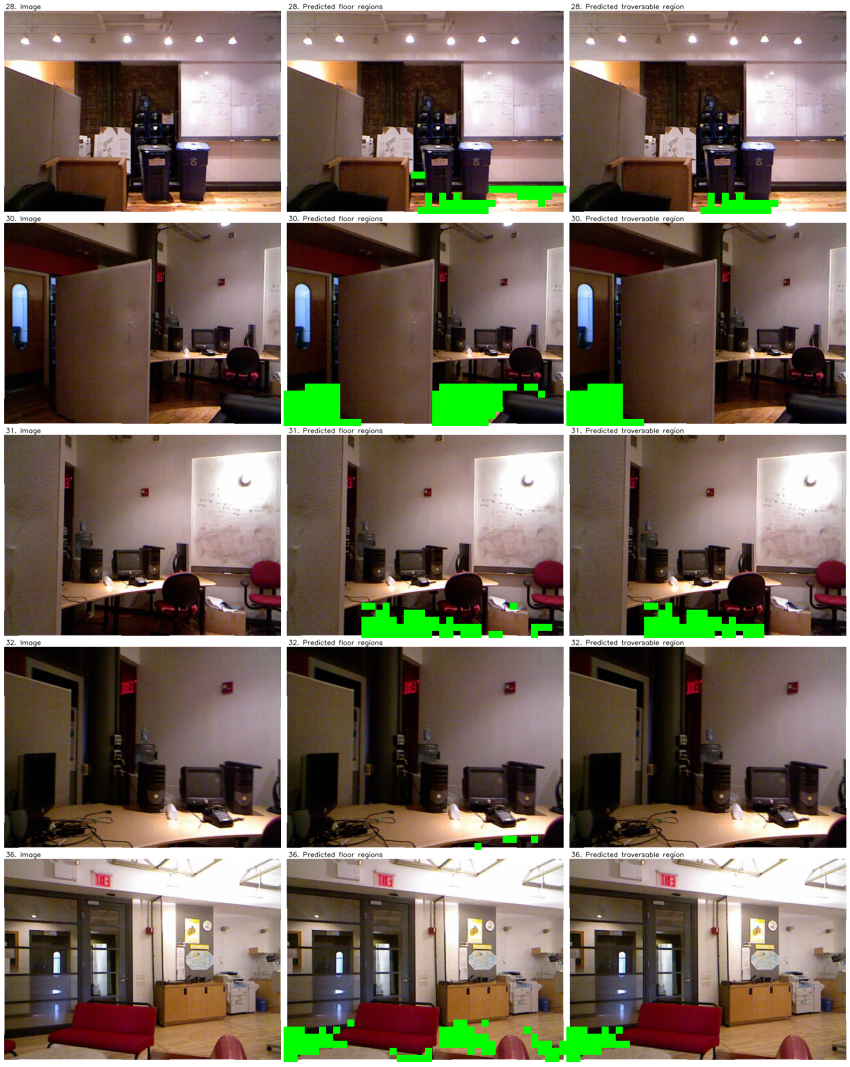


Figure 5: Examples of detection of closest-to-user traversable region. Each row corresponds to a different scene. First column shows the RGB image. Second column presents the predicted floor regions in green. Last column shows the identified closest-to-user traversable region in green.

(ASNS) for mobile robots. *Intelligent Service Robotics*, 9(3):187–205, 2016.

[2] A. Aladren, G. Lopez-Nicolas, L. Puig, and J.J. Guerrero. Navigation assistance for the visually impaired using rgb-d sensor with range expansion. *IEEE Systems Journal*, 10(3):922–932, 2016.

[3] J. Bai, Z. Liu, Y. Lin, Y. Li, S. Lian, and D. Liu. Wearable travel aid for environment perception and navigation of visually impaired people. *Electronics*, 8(6), 2019.

[4] T. Barreira, D. A Rowe, M. Kang, and others. Parameters of walking and jogging in healthy young adults. *International Journal of Exercise Science*, 3(1):10, 2010.

[5] Shripad Bhatlawande, Manjunatha Mahadevappa, Jayanta Mukherjee, Mukul Biswas, Debabrata Das, and Somedeb Gupta. Design, development, and clinical evaluation of the electronic mobility cane for vision rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(6):1148–1159, 2014.

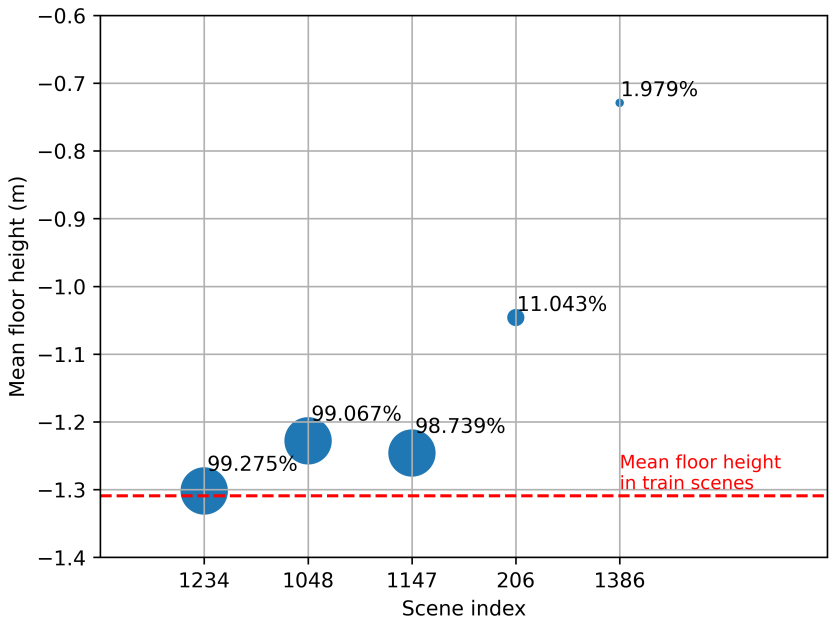


Figure 6: Relation between ground class accuracy and mean floor height in selected scenes of NYU-v2 dataset.

[6] C. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler. Convolutional patch networks with spatial prior for road detection and urban scene understanding.

[7] Erzhuo Che, Jaehoon Jung, and Michael J. Olsen. Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors*, 19(4), 2019.

[8] I. Goodfellow, Y. Bengio, and A Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] K. Imai, I. Kitahara, and Y. Kameda. Detecting walkable plane areas by using rgb-d camera and accelerometer for visually impaired people. volume 2017-June, pages 1–4, 2018.

[10] International Code Council. Means of Egress. In *International Building Code*, chapter 10. International Code Council, USA, Falls Church, Va, 2018.

[11] H. Jing and Y. Suya. Point cloud labeling using 3d convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670–2675, Dec 2016.

[12] S Khan, S. Rahmani, S.A.A Shah, M. Bennamoun, G. Medioni, and S. Dickinson. *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool Publishers, 2018.

[13] Y. Kida, S. Kagami, T. Nakata, M. Kouchi, and H. Mizoguchi. Human finding and body property estimation by using floor segmentation and 3d labelling. volume 3, pages 2924–2929, 2004.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[15] A. Leone, G. Diraco, and P. Siciliano. An automated active vision system for fall detection and posture analysis in ambient assisted living applications. pages 2301–2306, 2010.

[16] B. Li, J.P. Munoz, X. Rong, Q. Chen, J. Xiao, Y. Tian, A. Arditi, and M. Yousuf. Vision-based mobile indoor assistive navigation aid for blind people. *IEEE Transactions on Mobile Computing*, 18(3):702–714, 2019.

[17] Z. Li, F. Song, B.C. Clark, D.R. Grooms, and C. Liu. A wearable device for indoor imminent danger detection and avoidance with region-based ground segmentation. *IEEE Access*, 8:184808–184821, 2020.

[18] Huayao Liu, Ruiping Liu, Kailun Yang, Jiaming Zhang, Kunyu Peng, and Rainer Stiefelhagen. Hida: Towards holistic indoor understanding for the visually impaired via semantic instance segmentation with a wearable solid-state lidar sensor. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1780–1790, 2021.

[19] S. Liu, M. Zhang, P. Kadam, and C.C.J. Kuo. *3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods*. Springer International Publishing, 2021.

[20] H. Macher, T. Landes, and P. Grussenmeyer. Point clouds segmentation as base for as-built bim creation. volume 2, pages 191–197. Copernicus GmbH, 2015.

[21] S. Murakami, M. Shimakawa, K. Kivota, and T. Kato. Study on stairs detection using rgb-depth images. pages 1186–1191. Institute of Electrical and Electronics Engineers Inc., 2014.

[22] B. Nagy and C. Benedek. 3d cnn-based semantic labeling approach for mobile laser scanning data. *IEEE Sensors Journal*, 19(21):10034–10045, Nov 2019.

[23] M. Nakagawa and T. Kobayashi. Real-time floor recognition in indoor environments using tof camera. volume 1, pages 399–402. Asian Association on Remote Sensing, 2016.

[24] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[25] S. Öztürk, U. Özkaya, B. Akdemir, and L. Seyfi. Convolution kernel size effect on convolutional neural network in histopathological image processing applications. In *2018 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*, pages 1–5, 2018.

[26] J. Pardeiro, J.V. Gómez, D. Álvarez, and L. Moreno. Learning-based floor segmentation and reconstruction. *Advances in Intelligent Systems and Computing*, 253:307–320, 2014.

[27] R.P. Saputra and P. Kormushev. Casualty detection for mobile rescue robots via ground-projected point clouds. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10965 LNAI:473–475, 2018.

[28] H. Schulz, N. Höft, and S. Behnke. Depth and height aware semantic rgb-d perception with convolutional neural networks. In *Proc. Eur. Conf. Neural Netw.(ESANN)*, pages 463–468, 2015.

[29] Sicara. About convolutional layer and convolution kernel, 2018.

[30] Stanford University. Convolutional neural networks (cnns / convnets), 2020.

[31] Ruxandra Tapu, Bogdan Mocanu, and Titus Zaharia. Wearable assistive devices for visually impaired: A state of the art survey. *Pattern Recognition Letters*, 137:37–52, 2020. Learning and Recognition for Assistive Computer Vision.

[32] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017.

[33] I. Van Crombrugge, L. Mertens, and R. Penne. Fast free floor detection for range cameras. volume 4, pages 509–516. SciTePress, 2017.

[34] L. Wang, Z. Zhou, J. Wu, Y. Liu, and X. Zhao. Support-plane estimation for floor detection to understand regions' spatial organization. pages 2576–2581. Institute of Electrical and Electronics Engineers Inc., 2014.

[35] D. Wolf, J. Prankl, and M. Vincze. Enhancing semantic segmentation for robotics: The power of 3-d entangled forests. *IEEE Robotics and Automation Letters*, 1(1):49–56, 2016.

[36] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. Linking points with labels in 3d: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):38–59, 2020.

[37] K. Yang, K. Wang, R. Cheng, and X. Zhu. A new approach of point cloud processing and scene segmentation for guiding the visually impaired. 2015.

[38] C. Zatout, S. Larabi, I. Mendili, and S. Ablam Edoh Barnabe. Ego-semantic labeling of scene from depth image for visually impaired and blind people. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[39] Jiaying Zhang, Xiaoli Zhao, Zheng Chen, and Zhejun Lu. A review of deep learning-based semantic segmentation for point cloud. *IEEE Access*, 7:179118–179133, 2019.

A Results achieved by different configurations of the proposed CNN.

Input shape	Filters	Neurons	Accuracy				
			Ground	Struct	Furnit.	Props	Balanced
(8, 8, 3)	16	16	0.924805	0.752441	0.630859	0.221558	0.632324
(8, 8, 3)	16	32	0.923828	0.75	0.628418	0.222168	0.630859
(8, 8, 3)	16	64	0.924316	0.746582	0.637207	0.222168	0.632812
(8, 8, 3)	32	16	0.921875	0.746582	0.634277	0.225708	0.632324
(8, 8, 3)	32	32	0.921387	0.750488	0.633301	0.221802	0.631836
(8, 8, 3)	32	64	0.919434	0.749512	0.637695	0.217773	0.630859
(8, 8, 3)	64	16	0.92041	0.749023	0.638184	0.217529	0.631348
(8, 8, 3)	64	32	0.921387	0.750488	0.636719	0.216187	0.631348
(8, 8, 3)	64	64	0.920898	0.753418	0.637695	0.211548	0.630859
(16, 16, 3)	16	16	0.928711	0.761719	0.571777	0.288574	0.637695
(16, 16, 3)	16	32	0.932617	0.750977	0.576172	**0.294678**	**0.638672**
(16, 16, 3)	16	64	**0.935059**	0.752441	0.587402	0.277832	0.638184
(16, 16, 3)	32	16	0.920898	0.747559	0.609375	0.270996	0.637207
(16, 16, 3)	32	32	0.90918	0.760254	0.629395	0.227173	0.631348
(16, 16, 3)	32	64	0.914551	0.74707	0.628906	0.254395	0.63623
(16, 16, 3)	64	16	0.921387	0.737305	0.620605	0.273682	0.638184
(16, 16, 3)	64	32	0.919922	0.750977	0.635742	0.233765	0.635254
(16, 16, 3)	64	64	0.918457	0.754883	**0.643555**	0.217651	0.633789
(32, 32, 3)	16	16	0.911621	**0.833984**	0.500977	0.170288	0.604004
(32, 32, 3)	16	32	0.913574	0.802734	0.59375	0.105347	0.604004
(32, 32, 3)	16	64	0.904297	0.797363	0.587891	0.15332	0.61084
(32, 32, 3)	32	16	0.908203	0.800293	0.569824	0.161865	0.609863
(32, 32, 3)	32	32	0.90625	0.79541	0.562012	0.189209	0.613281
(32, 32, 3)	32	64	0.908203	0.792969	0.578125	0.174683	0.613281
(32, 32, 3)	64	16	0.910156	0.796875	0.570801	0.174316	0.613281
(32, 32, 3)	64	32	0.911133	0.793457	0.583008	0.163818	0.612793
(32, 32, 3)	64	64	0.911133	0.791504	0.592285	0.157349	0.613281
(64, 64, 3)	16	16	0.897461	0.750488	0.508301	0.215454	0.592773
(64, 64, 3)	16	32	0.894043	0.765625	0.519043	0.19751	0.594238
(64, 64, 3)	16	64	0.887695	0.761719	0.521484	0.206299	0.594238
(64, 64, 3)	32	16	0.890625	0.759766	0.533203	0.194458	0.594727
(64, 64, 3)	32	32	0.888184	0.763184	0.53418	0.191772	0.594238
(64, 64, 3)	32	64	0.890137	0.762695	0.538574	0.182739	0.59375
(64, 64, 3)	64	16	0.890625	0.753906	0.52832	0.200562	0.593262
(64, 64, 3)	64	32	0.888184	0.759277	0.536133	0.179321	0.59082
(64, 64, 3)	64	64	0.885254	0.746094	0.539551	0.190552	0.590332

B Results achieved by model 16-16-64 using different values of batch size and learning rate.

Batch size	Learning rate	Accuracy				
		Ground	Struct	Furnit.	Props	Balanced
32	0.001	**0.939453**	**0.751953**	0.603027	0.277588	**0.643066**
32	0.0001	0.929199	0.705566	0.580078	**0.307373**	0.630371
64	0.001	0.922852	0.727051	**0.603516**	0.258789	0.62793
64	0.0001	0.928223	0.708008	0.603516	0.269531	0.627441
128	0.001	0.928711	0.720703	0.595215	0.268311	0.628418
128	0.0001	0.931152	0.703613	0.591309	0.282715	0.626953
256	0.001	0.929199	0.713379	0.591309	0.270264	0.625977
256	0.0001	0.931152	0.700195	0.586914	0.283203	0.625488
512	0.001	0.931152	0.704102	0.592773	0.280518	0.626953
512	0.0001	0.932617	0.693848	0.589355	0.289307	0.626465