

Mutual use of semantics and geometry for CNN-based object localization in ToF images

Antoine Vanderschueren*, Victor Joos* and Christophe De Vleeschouwer

ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium

{antoine.vanderschueren, victor.joos,
christophe.devleeschouwer}@uclouvain.be

*Authors contributed equally to the paper

Abstract. We propose a novel approach to localize a 3D object from the intensity and depth information images provided by a Time-of-Flight (ToF) sensor. Our method builds on two convolutional neural networks (CNNs). The first one uses raw depth and intensity images as input, to segment the floor pixels, from which the extrinsic parameters of the camera are estimated. The second CNN is in charge of segmenting the object-of-interest so as to align its point cloud with a reference model. As a main innovation, the object segmentation exploits the calibration estimated from the prediction of the first CNN to represent the geometric depth information in a coordinate system that is attached to the ground, and is thus independent of the camera elevation. In practice, both the height of pixels with respect to the ground, and the orientation of normals to the point cloud are provided as input to the second CNN. Our experiments, dealing with bed localization in nursing homes and hospitals, demonstrate that our proposed floor-aware approach improves segmentation and localization accuracy by a significant margin compared to a conventional CNN architecture, ignoring calibration and height maps, but also compared to PointNet++.

Keywords: Depth · CNN · Object Localization.

1 Introduction

In hospitals and nursing homes, the number of nurses at night is largely insufficient to keep a permanent eye on every patient or senior. Automatic human behavior analysis is therefore required to help with the detection of bed exits and falls, to alert the medical staff as soon as possible.

In this context, the Time-of-Flight (ToF) camera offers the following non-negligible advantages: it provides a depth map, in addition to the reflected intensity, and, given its active nature, is relatively independent of lighting conditions. These advantages come with greatly reduced image resolution, and a shorter range. Hence, ToFs appear especially suited for the monitoring of small closed spaces like bedrooms and hospital rooms, especially at night-time. Detection of humans, often represented as moving blobs, from ToF image sequences has been largely investigated [10, 25, 28]. However, turning those detections into

human behavior interpretation requires to position the camera with respect to the scene, and to localize the objects the human interacts with. This preliminary but critical step is often neglected, assuming that calibration and scene composition is encoded manually. An autonomous calibration would however facilitate the deployment of systems in real-life conditions, and give the opportunity to adjust the interpretation of movements to the displacement of key objects in the scene.

Our work focuses on this calibration step, and investigates a use case that aims at localizing the beds in rooms of nursing homes or hospitals. Beyond the scope of this paper, this would typically be combined with human detection, for bed exit and/or fall recognition.

In short, our approach to automatically calibrate the camera and position the bed in a room, follows the steps illustrated in Fig. 1. It first segments the floor based on a CNN fed with raw depth and intensity maps. This initial segmentation allows us to estimate the ground plane equation in the coordinate system of the camera, so that the depth information can be transformed to height information with respect to the floor.

The second step of our method consists in segmenting the object-of-interest by feeding a second CNN with the ToF intensity map, the height information, and the field of vectors defining the local normals to the point cloud. The segmented point cloud is then aligned with a model of the object, to localize the object in the scene. Our method is validated on a *practical* case using *real* data. This case considers the localization of beds in nursing homes and hospital rooms. Our study results in a multifold contribution. Specifically, our work offers:

- Precise and automatic estimation of the floor plane position in the ToF camera referential;
- Effective segmentation of the point cloud associated to the object of interest, here the bed, using a 2D CNN. The approach is shown to outperform networks operating directly on point clouds like PointNet++ [22]. The method is also shown to improve in accuracy when the network is fed with geometric information represented in terms of *height* (with respect to the floor) and *local normals* (defined in a referential aligned with the floor normal);
- Fast and reliable localization of the object of interest, thanks to the floor plane knowledge, which reduces the degrees of freedom from 6 to 3 when aligning the reference model with the segmented point cloud;
- The estimation of localization confidence, enabling the system to wait for better observation conditions (no occlusions, see Fig. 3a and 3b) or for human intervention.

This article is organized in 3 main sections. Section 2 surveys the state-of-the-art related to object segmentation and localization. Section 3 then describes our method, and discusses its strengths and weaknesses compared to previous works. Section 4 validates our method on a real-life use case.

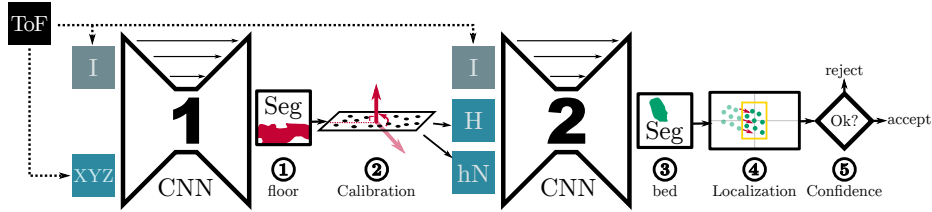


Fig. 1. Overview of our method. (1) The first CNN segments the floor based on the intensity map (I) and spatial coordinates (XYZ) in the camera-centric viewpoint of the ToF. (2) The normal to the ground plane is estimated from the segmented 3D floor points, and is used to define a height map (H) and a field of local normals (hN) represented in a referential obtained by rotating the ToF referential to make its Z axis orthogonal to the ground plane. (3) The second CNN combines these (H) and (hN) maps with the intensity map to segment the object-of-interest. (4) Segmented points are aligned with the reference model, to localize the object. (5) A level of confidence is assigned to the localization, by comparing the point cloud and the fitted model.

2 State of the Art

The literature addressing object localization based on depth information (either from RGB-D or ToF) considers two distinct methodologies. Some recent efforts, similar to our works, adopt a CNN to identify the pixels belonging to the object, and use the corresponding 3D points to compute the object position (Section 2.1). Others directly process a 3D point cloud, using CNNs or graphs (Sections 2.2 and 2.3). Using 2D segmentation, we identify the pixels belonging to the object, and use the corresponding 3D points to compute the object position.

2.1 2D Convolutional Segmentation

The similarity between the signal, output by RGB-D and ToF makes it relevant to extend the quite laconic SotA related to ToF segmentation [6, 16], to the broader literature related to RGB-D segmentation [2].

The segmentation methods using RGB-D signals as inputs of 2D CNNs differ in the way they merge the color and depth signals. This fusion generally depends on the chosen network architecture [4, 11], and is sometimes even driven by a squeeze-and-excite attention module [8, 9].

However, our experiments (not presented here for conciseness) have revealed that in the case of ToF data, there is no benefit to fusing Intensity and Depth based on attention modules. We have therefore devised a straightforward and computationally simpler fusion, described in Section 3.3.

A fundamental difference between our work and previous art lies in the way the depth signal is represented to feed the CNN. Based on the automatic ground/floor plane parameters estimation, the depth is transformed into a height value for each image pixel. Moreover the neighborhood of each 3D point is used to compute a local normal to the point cloud, described in a referential aligned

with the floor normal. We show that this original representation improves the CNN accuracy. This confirms the results reported in [3] regarding the use of a geocentric representation, which encodes height above ground and angle with gravity for each pixel, for the detection of objects with a pre-calibrated RGB-D system combining CNN and SVM.

2.2 3D Convolutional Localization

3D Convolutional neural networks build on a voxelization of the point cloud. They suffer either from the lack of resolution induced by the use of big voxels, or from high sparsity in voxel information.

3D networks have first been considered for object localization in [17]. This pioneering work uses a U-Net [23] structure (like the one used in this paper), but substituting 2D for 3D convolutions. Quite recently, [7] has proposed to combine the high-resolution 2D color information, from RGB-D data, with a 3D neural network on point cloud data, leading to a precision gain of 2-3%. In our applicative context, we show in Section 4.4 that ToF intensity images contribute the *least* of all the input types, to the final segmentation decision.

[26] has trained a network on synthetic data to complete the voxels that remain hidden when a single viewpoint is available. We have however observed that networks trained on synthetic data did not transfer well to real-life ToF data.

2.3 Segmentation and Localization using Graph NNs

To circumvent the excessive computational cost of 3D convolutions dealing with high voxel resolution, graph neural networks work on connected points rather than on regular 2D or 3D matrices. Most implementations combine fully-connected layers and specialized pooling layers. Most point-based segmentation [13, 14] and object localization [12, 19, 20, 24] methods are based in part or in full on PointNet [21] and PointNet++ [22]. Those approaches, although attractive in the way they represent the input point cloud, still show a lack of accuracy compared to convolutional methods on RGB-D and ToF images.

3 Floor plan estimation for Autonomous Object Localization

To describe our method, we start by explaining the structure of our framework, and then delve deeper in its building blocks.

3.1 ToF Data

The ToF sensor gives us, for every pixel, information about reflected intensity and depth, i.e. distance to the camera. Having access to the intrinsic parameters of the camera, we can express depth in terms of 3D position relative to the camera (XYZ). We can also estimate the normal vector to the point cloud surface in each point in this same coordinate system.

3.2 Method Overview

Fig. 1 depicts the five steps of our method:

Step 1. A first CNN uses the intensity and depth data to segment the floor pixels.

Step 2. The floor’s plane equation in 3D space, relative to the camera, is obtained via Singular Value Decomposition (SVD) on the 3D coordinates of the floor pixels. The vector that is normal to the ground plane is defined by the smallest singular value. To make our algorithm more robust towards outliers, we embed SVD into a RANdom SAMpling Consensus (RANSAC) algorithm.

Step 3. A second CNN segments the object-of-interest. Since the ground plane equation is known, the height of every pixel can be computed from its XYZ coordinates. We consider two alternatives to feed the neural network with floor-aware geometrical information.

1. In the first alternative, the resulting pixel height map (H) is fed to the network, together with the 3 components of the normal (hN) to the surface point cloud in each point. Both H and hN are expressed in a referential obtained by rotating the camera referential (first around Z, originally pointing towards the scene, to make X horizontal, and then around the X-axis) to align the Z-axis with the floor normal.
2. In the 2nd alternative, since the floor plane equation has been estimated, the intensity (I) and height (H) information associated to the 3D point cloud can be projected on the floor plane (see Fig. 6a). This provides a bird’s-eye 2D view of the scene, which is independent of the actual camera elevation (up to occlusions). We have trained a network to predict the bed label from those two projections, respectively denoted $\text{proj}(I)$ and $\text{proj}(H)$, taken as inputs.

Step 4. The coordinates of the segmented object’s 3D points are then fed into a localization algorithm that aligns them with a reference model.

Step 5. The quality of the matching between the object points and the aligned model is used as a localization confidence score, to detect cases of heavy occlusion or incorrect segmentation, as shown in Fig. 3a and 3b.

It should be noted that the calibration step (the 2nd step in the list above) allows us to move away from a camera-centric coordinate system towards a *floor-centric* point-of-view. Hence, instead of expressing the data in a coordinate system that is fully dependent on the camera placement, we express them in a referential that is independent of the camera elevation. This offers the two following benefits:

- The object segmentation network learns the relationship between the geometric information and the object’s mask more easily (as we’ll show quantitatively and qualitatively in Section 4)
- The degrees of freedom to consider during the localization step get reduced from 6 (3 rotations, 3 translations) to 3 (1 rotation, 2 translations).

The remainder of this section, presents how the CNNs used in steps 1 and 3, respectively for floor and object segmentation, have been constructed and trained. It also describes the implementation details for the localization step.

3.3 Floor and Object Segmentation (steps 1 and 3)

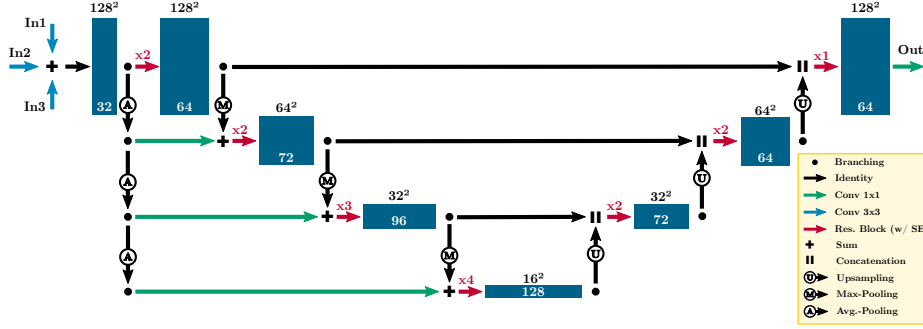


Fig. 2. Our MultiNet Network Architecture. The arrows depict convolutional layers. When present, the multiplicative factor along the arrow defines the number of times the convolutional block is repeated. The boxes represent the feature maps, the number of channels are noted inside the box, the resolution outside. The architecture follows the U-Net model, with the addition of multi-resolution inputs, and block repetitions as shown in the figure. The different inputs are fused using a sum after an initial convolutional block. This architecture is used for both floor and object segmentation.

To segment the point cloud, we use the U-Net [23] shaped network presented in Fig. 2, and denoted *MultiNet* in the rest of the paper, since it is suited to handle multiple types of inputs. U-Net adapts an auto-encoder structure by adding skip-connections, which link feature maps of identical resolution from the encoder to the decoder. This allows the direct transfer of high resolution information to the decoding part, by avoiding the lower resolution network bottleneck.

Feature maps of identical resolution are said to be of the same *level*. Each level's structure, is based on residual blocks [5] followed by a *Squeeze-and-Excite* module that weighs every feature map individually before their sum [8].

The convolutional blocks are repeated, as indicated by the multiplicative factors along the arrows in Fig. 2. Those repetition factors follow the parameters recommended in [1, 11, 29].

Our networks are fed, as explained in the method overview, by a combination of the following input types: intensity (1D), normal (3D), XYZ (3D), and height (1D). To deal with different types of inputs we merge all inputs directly : every input type is passed once through a convolutional layer, such that they all possess the 32 feature maps. These feature maps are then summed once and fed to the rest of the network at every level, through down-scaling.

3.4 Localization and error estimation (steps 4 and 5)

As explained in the method overview, the pixels labeled by our segmentation network as being part of the object, i.e. the bed, are used to localize it in a coordinate system where one axis is perpendicular to the floor, and the two remaining axes are respectively parallel and perpendicular to the intersection of the ToF image plane with the ground plane.

Since the bed has a simple shape we adopt a very basic localization approach on a rasterized (discretized on a 5cm^2 resolution grid) 2D projection of the segmented points on the ground plane.

As shown in Fig. 3c, a rectangular shape is considered to model the bed. The center of mass and the principal direction, estimated via SVD, of the projected points are used to initialize the model alignment process. This procedure consists in a local grid search and selects the model maximizing the estimated intersection-over-union, denoted rIoU^b , between the rasterized projected points (aggregating close points) and the 2D rectangular shape defined by the searched parameters.

Since it reflects the adequacy between the selected rectangular model and the projected point cloud, the rIoU^b of the 2D rectangular model is then used as a confidence score to validate or reject the predicted bed localization (5th step). It can be used to detect a wrong prediction, e.g. occurring when a bed is occluded by a nurse, and repeat the process in better observation conditions.

4 Results and Analysis

This section first introduces our validation methodology, including use-case definition, training strategy, and quantitative metrics used for evaluation. It then considers floor and object segmentation, respectively in Sections 4.2 and 4.3. Eventually, Section 4.4 considers an ablation study to assess the benefit resulting from our proposed representation of the geometric information.

4.1 Validation methodology

Use case Our method is evaluated on a ToF dataset, captured in nursing homes and hospital rooms. Our objective is to position the bed in the room, without any additional information but the intrinsic parameters of the camera and the images it captures. To assess the performance of our system when it is faced with a new room layout or style, we divide our dataset in 7 subsets containing *strictly* different institutions (hospitals or nursing homes). We apply cross-validation in order to systematically test the models on rooms that have not been used during training.

Our database contains 3892 images of resolution 160×120 . Those images come from 85 rooms belonging to 11 institutions. On average, 45 images with divers illumination, occlusions and (sometimes) bed positions are available per room.

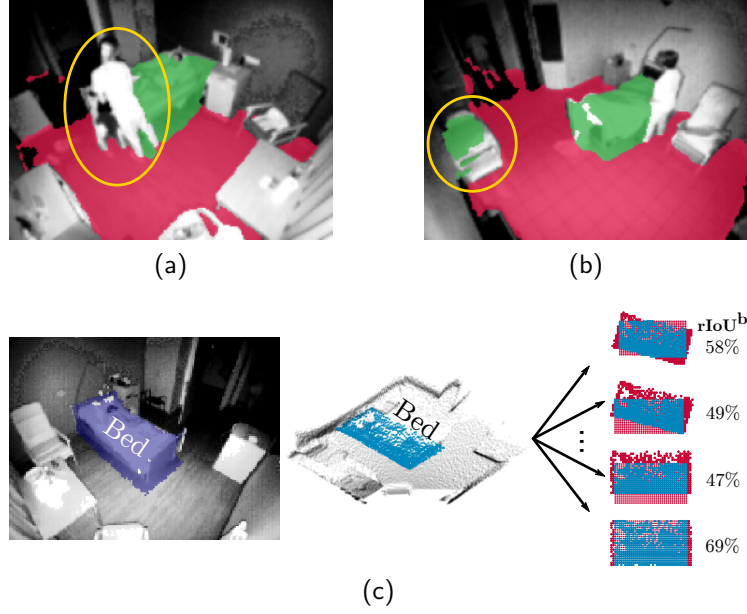


Fig. 3. (a) Example of occlusion. (b) Example of wrong segmentation. (c) Example of localization and error estimation. After the object segmentation, the image pixels are projected onto the 2D plane of the floor. The points that were segmented as “bed” are then extracted, and various transformations (2D translation and 1D rotation) are tried until the final bounding box is found with the maximization of the IoU on a discretized grid ($rIoU^b$) between prediction and model (blue and small red points). The intersection contains only the blue points, while the union contains blue and red points (of all sizes). *Best viewed in color.*

In order to train and validate our models, we use manually annotated data, both for the device calibration and for the localization of the bed.¹

Training During our cross-validation, we select 1 subset for testing, 1 subset for validation and 5 for training. Cross-validation on the test set is used due to the small number of different. We use data augmentation in the form of vertical flips and random zooming. The images are then cropped and rescaled to fit a 128x128 network input resolution.

Normal vectors are estimated from the 10 closest neighbors that are within a 10 cm radius of each point. They are expressed in a floor-aware referential (hN) as detailed in Section 3.2.

All our segmentation networks result from a hyper-parameter search evaluated on each validation set. The average performance of all 7 subsets obtained

¹ The tool developed for annotation is available at <https://github.com/ispgroupucl/tofLabelImg>.

on the test set is then presented. The networks are trained using AdamW [15], using a grid search to select the learning rate and the weight decay in $\{1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4}\}$ and $\{1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6}\}$ respectively. The learning rate is divided by 10 at epochs 23 and 40, while training lasts for 80 epochs, with a fixed batch size of 32. We implement our segmentation models using PyTorch [18] and have published our code at <https://github.com/ispgroupucl/tof2net>. For the PointNet++ segmentation model, we use the model described in [22] and implemented by [27]. We never start from a pre-trained network as pre-training shows poor performance on our dataset in all cases.

Finally in order to avoid initialization biases, the values presented in *every* table are always the average of 5 different runs.

Metrics Distinct metrics are used to assess segmentation, calibration and bed localization.

Segmentation predictions are evaluated using Intersection-over-Union between predicted and ground-truth pixels. This metric is denoted IoU. Recall ($\frac{TP}{TP+FN}$) and precision ($\frac{TP}{TP+FP}$) are also considered separately, to better understand the nature of segmentation failures.

We evaluate the extrinsic camera calibration using absolute angles between the ground normal predicted by our model and the ground truth.

Object localization is also evaluated using Intersection-over-Union, but instead of comparing sets of pixels, we compute the intersection-over-union between the 2D bounding-box obtained after localization and the ground truth, projected on the estimated floor plane. This metric is denoted IoU^b , with b referring to the fact that bounding boxes are compared. The projection on a common plane is necessary to compensate for slight differences due to possible calibration errors i.e. different ground plane equations.

In practice, IoU^b that lie below 70% correspond to localization not sufficiently accurate to support automatic behavior analysis, typically to detect when a patient leaves the bed. Hence we consider 70% as a relevant localization quality threshold, and evaluate our methods based on the Average Precision at this threshold (AP@.7). In addition, the Area Under Curve (AUC@.7) measures the correct localization predictions (true positives) as a function of the incorrect ones (false positives) when scanning the confidence score given by the estimated rIoU^b , explained in Section 3.4.

4.2 Floor segmentation and floor normal estimation

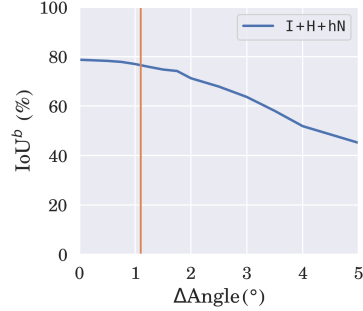
Table 1 compares the different floor segmentation models in terms of segmentation and floor normal accuracy. It also presents the floor normal estimation error obtained when applying RANSAC directly on the whole set of points (since a majority of points are floor points, one might expect RANSAC will discard outliers and estimate the ground plane equation without segmentation). We observed that the global RANSAC performance is not sufficient for localization purposes. With a mean error greater than 10° it would lead to a drop in IoU^b of more than 40%, as shown in Fig. 4.

Table 1. Floor Segmentation and Calibration Results.

Method	Inputs ¹	IoU (%)	ΔAngle (°)
RANSAC	XYZ	-	13.2
PointNet++	XYZ+I	83.6	1.2
MultiNet	I	75.6	3.3
MultiNet	I+XYZ	87.2	1.1

Best results in **bold**

¹ I denotes intensity, XYZ defines the 3D points spatial coordinates in the ToF camera referential.

**Fig. 4.** Bed Localization bounding-box Intersection-over-Union (IoU^b) dropoff as a function of the floor normal direction estimation error. The orange line is at 1.1° , which corresponds to the mean error from MultiNet-I+XYZ.

Looking at the segmentation IoUs, we see that PointNet++ has better accuracy than a convolutional network that only uses the intensity (I) as input signal. However, adding the XYZ point coordinates (as defined in the coordinate system of the camera) as input to the convolutional model is enough to surpass PointNet++.

In terms of ground normal estimation, our proposed model leads to a mean absolute error of 1.1° . As can be seen in Fig. 4 this is precise enough for our use-case. Indeed, the segmentation quality of our floor-aware CNNs (see Section 4.3) remains nearly constant for calibration errors lower than 1° . For errors greater than 1.5 - 2° a substantial performance loss is observed. MultiNet-I+XYZ however has a mean error of 1.1° positioning our approach as an acceptable solution that won't lead to a significant error propagation. We also note that even though the segmentation maps made by PointNet++ is significantly worse than the ones predicted by MultiNet-I+XYZ, the final floor normal error is only slightly worse than MultiNet-I+XYZ.

4.3 Bed segmentation and localization accuracy

Table 2 summarizes the results of bed segmentation and localization for different representations of the geometric information.

Segmentation The object localization relies heavily on the object segmentation accuracy. For this reason we first compare the different methods in terms of segmentation in the third column of Table 2. For all the MultiNets using the H and/or hN representation of the geometric information, the floor calibration is the one produced with the MultiNet-I+XYZ from Section 4.3, thus possible calibration errors have been propagated to the final numerical values.

Table 2. Bed Segmentation and Localization Results

Method	Inputs ¹	IoU(%)	IoU ^b (%)	AP@.7	AUC@.7
PointNet++ [22]	XYZ+I	44.6	60.8	46.2	43.5
MultiNet	I	65.3	67.1	60.0	58.6
MultiNet	I+XYZ	69.9	75.4	76.2	74.7
MultiNet	I+H	<u>71.0</u>	76.1	78.5	77.1
MultiNet	I+H+hN	72.1	<u>77.2</u>	<u>80.6</u>	<u>79.1</u>
MultiNet	proj(I)+proj(H)	N/A	78.1	83.1	81.8

Best results in **bold**, 2nd best underlined

¹ I denotes intensity, XYZ spatial coordinates, H height, and hN the local normals, defined in the floor-aligned referential, proj(X) the projection of X onto the floor-aligned plane.

We were initially surprised by the very low accuracy of PointNet++, even compared to the intensity-only MultiNet baseline. However, this can be explained by the fact that the shape of the bed is more complex than the planar floor geometry. In addition, ToF cameras are known to induce large measurement disparities, making it harder for a geometry-based neural network to correctly learn the object’s geometry.

Looking at our MultiNet methods, we see that accounting for the geometric information systematically outperforms the single intensity baseline. Moreover, our proposed representation of geometry in terms of H and hN improves the conventional XYZ representation by more than 2%.

Fig. 5 shows qualitative results. The results show the advantage of our floor-aware approach (third column).

Localization Table 2 also displays the metrics for bed localization. When looking at the bounding-box IoU^b or the average precision at a threshold of 0.7 IoU^b (AP@.7), we observe the same trends as with the segmentation IoU. The gap between a formulation with and without floor calibration information is however reduced to 1.8% for IoU^b but widened to 4.4% for AP@.7 and AUC@.7. The results obtained when feeding the network with the floor-plane projected intensity (proj(I)) and height (proj(H)) information of each 3D point are shown in the last row of Table 2. The average quality of the localization increases by another 0.9% compared to MultiNet+I+H+hN. However, when looking at Fig. 6b, the higher medians of I+XYZ and I+H+hN reflect a higher number of bad bed localizations, but also a better bed localization whenever the viewing conditions are ideal.

Finally, we consider the relation between AP@.7 and AUC@.7 to evaluate the relevance of our confidence threshold rIoU^b. AUC@.7 denotes the area-under-curve when plotting the percentage, compared to the whole set of samples, of correct (meaning with IoU^b > 0.7) localization as a function of the percentage of

samples above the confidence threshold, for a progressively increased confidence threshold. By definition, this value is upper bounded by $AP@.7$, and gets close to this upper bound when samples with correct (erroneous) localization correspond to high (low) $rIoU^b$ confidence values.

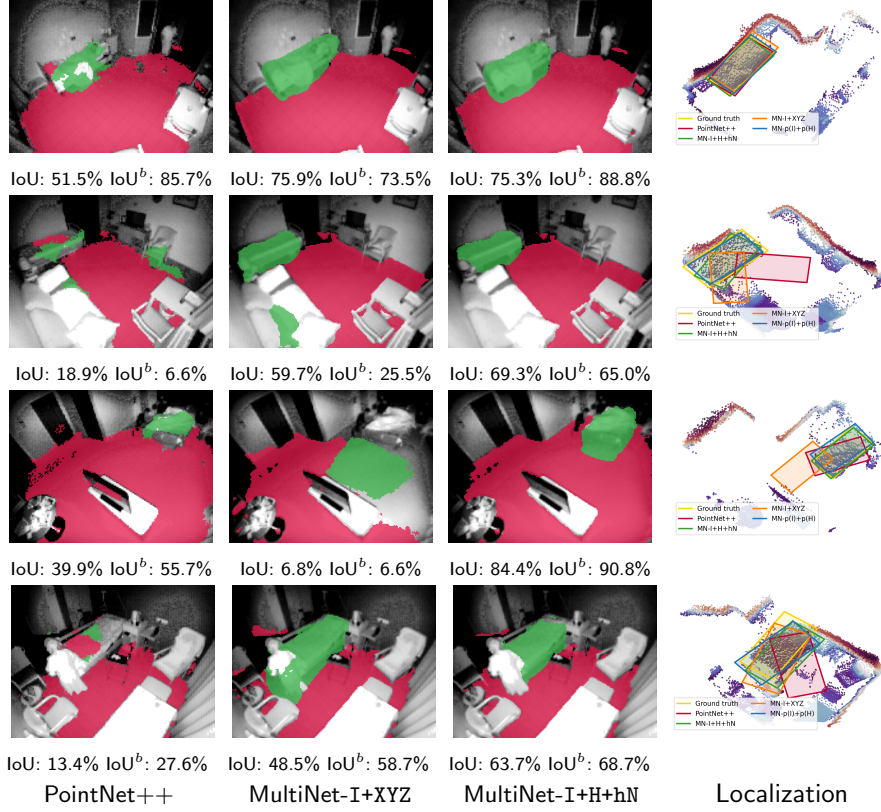


Fig. 5. Qualitative results of segmentation and localization. The first three columns show the segmentation results of PointNet++, MultiNet+I+XYZ, and MultiNet+I+H+hN. The last column shows the localization results for the same models in a top view with height encoded in point color. Floor points have been removed for easier visualization. PointNet++ has a tendency to correctly locate the bed in most cases, but is unable to fully segment the bed. MultiNet+I+XYZ shows correct segmentation in most cases, but has problems locating beds in complex locations, for example in the third row. MultiNet+I+XYZ also suffers from oversegmentation, as shown in the first and last rows. MultiNet+I+H+hN is able to segment most cases correctly, but can suffer from biased segmentation for the localization, as in row 2 and 4. The last column shows clearly that a correct segmentation leads to good localization. *Best viewed in color.*

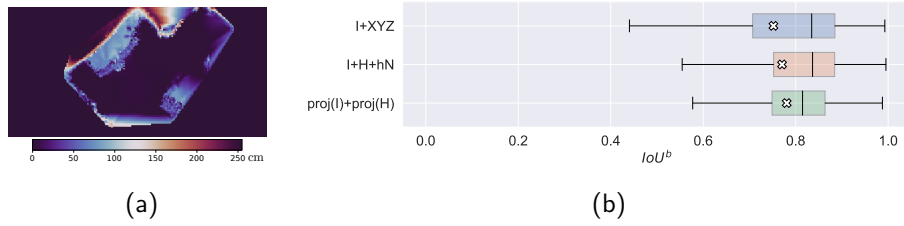


Fig. 6. (a) Example of $\text{proj}(\mathbf{H})$. Color indicating distance from floor. (b) Distribution of IoU^b for different models. From upper to lower: boxplot of the IoU^b for $\mathbf{I}+\mathbf{XYZ}$, $\mathbf{I}+\mathbf{H}+\mathbf{hN}$ and $\text{proj}(\mathbf{I})+\text{proj}(\mathbf{H})$. The white cross indicates the mean of each distribution. We can see that while the mean of the $\text{proj}(\mathbf{I})+\text{proj}(\mathbf{H})$ model is higher, its median and 75th percentile are lower.

Table 3. Ablation of the MultiNet trained on $\mathbf{I}+\mathbf{XYZ}+\mathbf{H}+\mathbf{N}+\mathbf{hN}$

	All	Independent removal of				
		I	N	XYZ	hN	H
IoU	70.6	67.1	66.0	63.4	57.3	27.6

I denotes intensity, XYZ spatial coordinates, H height, and N and hN local normals in the camera or floor-aligned referential.

4.4 Input ablation

In order to strengthen the intuition that geometric information, and the way it is presented is important, we now look at a network trained with every possible input type: $\mathbf{I}+\mathbf{XYZ}+\mathbf{N}+\mathbf{H}+\mathbf{hN}$. In that way, the network gets the opportunity to select its preferred representation of the spatial information.

The Direct Fusion, explained in more details in Section 3, is the summation of every input convolved once. Each input can thus individually be removed from the sum. Doing this outright leads to activation-range scaling issues since the network isn’t retrained. Thus every channel in the removed component is replaced by the mean of its activations. We do this for every input-type independently, and present the results in Table 3.

The network is severely penalized when removing hN or H, which reveals that it consistently prefers height-encoded information. This confirms our hypothesis that a floor-aware encoding of the geometric information provides a representation that is easier to digest by the CNN.

5 Conclusion

We propose a method that successfully localizes beds in hospital and nursing home rooms. Our method estimates the extrinsic parameters of the device in order to have access to height maps and to normal vectors encoded in a referential aligned with the floor normal. We extensively show that this way of encoding ToF data leads to better performing CNNs. We recommend future work on input analysis using both visualization and network-based input-importance learning.

References

1. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: *Lecture Notes in Computer Science*. vol. 11211 LNCS, pp. 833–851 (2018). https://doi.org/10.1007/978-3-030-01234-2_49
2. Fooladgar, F., Kasaei, S.: A survey on indoor RGB-D semantic segmentation: from hand-crafted features to deep convolutional neural networks. *Multimedia Tools & Applications* **79**, 4499–4524 (2020). <https://doi.org/10.1007/s11042-019-7684-3>
3. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. *Lecture Notes in Computer Science* **8695**(PART 7), 345–360 (2014). https://doi.org/10.1007/978-3-319-10584-0_23
4. Hazirbas, C., Ma, L., Domokos, C., Cremers, D.: FuseNet: Incorporating Depth into Semantic Segmentation via Fusion-Based CNN Architecture. *ACCV* pp. 213–228 (2016). https://doi.org/10.1007/978-3-319-54181-5_14
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. *CVPR* (2016). <https://doi.org/10.3389/fpsyg.2013.00124>
6. Holz, D., Schnabel, R., Droschel, D., Stückler, J., Behnke, S.: Towards semantic scene analysis with time-of-flight cameras. *Lecture Notes in Computer Science* **6556**(June), 121–132 (2011). https://doi.org/10.1007/978-3-642-20217-9_11
7. Hou, J., Dai, A., NieBner, M.: 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. *CVPR* pp. 4416–4425 (2019). <https://doi.org/10.1109/cvpr.2019.00455>
8. Hu, J., Shen, L., Sun, G.: Squeeze-and-Excitation Networks. *CVPR* pp. 7132–7141 (2018). <https://doi.org/10.1109/CVPR.2018.00745>
9. Hu, X., Yang, K., Fei, L., Wang, K.: ACNET: Attention Based Network to Exploit Complementary Features for RGBD Semantic Segmentation. *ICIP* pp. 1440–1444 (2019). <https://doi.org/10.1109/ICIP.2019.8803025>
10. Jia, L., Radke, R.J.: Using time-of-flight measurements for privacy-preserving tracking in a smart room. *IEEE Transactions on Industrial Informatics* **10**(1), 689–696 (2 2014). <https://doi.org/10.1109/TII.2013.2251892>
11. Jiang, J., Zheng, L., Luo, F., Zhang, Z.: RedNet: Residual Encoder-Decoder Network for indoor RGB-D Semantic Segmentation. *arXiv preprint arXiv:1806.01054* pp. 1–14 (2018), <http://arxiv.org/abs/1806.01054>
12. Landrieu, L., Simonovsky, M.: Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. *CVPR* pp. 4558–4567 (2018). <https://doi.org/10.1134/1.559035>, <http://arxiv.org/abs/1711.09869>
13. Li, G., Müller, M., Thabet, A., Ghanem, B.: DeepGCNs: Can GCNs Go as Deep as CNNs? *ICCV* (2019), <http://arxiv.org/abs/1904.03751>

14. Liang, Z., Yang, M., Deng, L., Wang, C., Wang, B.: Hierarchical depthwise graph convolutional neural network for 3D semantic segmentation of point clouds. Proceedings - IEEE International Conference on Robotics and Automation **2019-May**, 8152–8158 (2019). <https://doi.org/10.1109/ICRA.2019.8794052>
15. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. ICLR (2019)
16. Maddalena, L., Petrosino, A.: Background subtraction for moving object detection in RGBD data: A survey. Journal of Imaging **4**(5) (2018). <https://doi.org/10.3390/jimaging4050071>
17. Milletari, F., Navab, N., Ahmadi, S.A.: V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In: Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016. pp. 565–571 (2016). <https://doi.org/10.1109/3DV.2016.79>
18. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. NeurIPS (2019), <http://arxiv.org/abs/1912.01703>
19. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep Hough Voting for 3D Object Detection in Point Clouds. ICCV (4 2019), <http://arxiv.org/abs/1904.09664>
20. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum Point-Nets for 3D Object Detection from RGB-D Data. CVPR (2018). <https://doi.org/10.1109/CVPR.2018.00102>, <http://arxiv.org/abs/1711.08488>
21. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 2016 Fourth International Conference on 3D Vision (3DV) pp. 601–610 (2016). <https://doi.org/10.1109/3DV.2016.68>
22. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. Advances In Neural Information Processing Systems (2017). <https://doi.org/10.3109/13816819409056905>
23. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. Lecture Notes in Computer Science **9351**, 234–241 (2015). https://doi.org/10.1007/978-3-319-24574-4_28
24. Shi, S., Wang, X., Li, H.: PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In: CVPR. pp. 770–779. IEEE (6 2019). <https://doi.org/10.1109/CVPR.2019.00086>
25. Simon, C., Meessen, J., De Vleeschouwer, C.: Visual event recognition using decision trees. Multimedia Tools and Applications **50**(1), 95–121 (10 2010). <https://doi.org/10.1007/s11042-009-0364-y>
26. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic Scene Completion from a Single Depth Image. CVPR (2017). <https://doi.org/10.1109/CVPR.2017.28>, <http://arxiv.org/abs/1611.08974>
27. Wijmans, E.: PointNet++ Pytorch Implementation. Github (2018), <https://github.com/erikwijmans/Pointnet2.PyTorch>
28. Yang, L., Ren, Y., Zhang, W.: 3D depth image analysis for indoor fall detection of elderly people. Digital Communications and Networks **2**(1), 24–34 (2 2016). <https://doi.org/10.1016/j.dcan.2015.12.001>
29. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: BiSeNet: Bilateral segmentation network for real-time semantic segmentation. Lecture Notes in Computer Science **11217**, 334–349 (2018). https://doi.org/10.1007/978-3-030-01261-8_20