3D OBJECT DETECTION BY FEATURE AGGREGATION USING POINT CLOUD INFORMATION FOR FACTORY OF THE FUTURE

F. Negin, A.K. Aijazi, L. Trassoudaine, P. Checchin*

Institut Pascal, UMR 6602, Université Clermont Auvergne, CNRS, SIGMA Clermont, F-63000 Clermont-Ferrand, France
(fahrood.negin, ahmad-kamal.aijazi, laurent.trassoudaine, paul.checchin)@uca.fr

KEY WORDS: Object detection; Point cloud; Factory of the future

ABSTRACT:

Nowadays, object detection is considered as an unavoidable aspect that needs to be addressed in any robotic application, especially in industrial settings where robots and vehicles interact closely with humans and objects and therefore a high level of safety for workers and machines is required. This paper proposes an object detection framework suitable for automated vehicles in the factory of the future. It utilizes only point cloud information captured by LiDAR sensors. The system divides the point cloud into voxels and learns features from the calculated local patches. The aggregated feature samples are then used to iteratively train a classifier to recognize object classes. The framework is evaluated using a new synthetic 3D LiDAR dataset of objects that simulates large indoor point cloud scans of a factory model. It is also compared with other methods by evaluating on SUN RGB-D benchmark dataset. The evaluations reveal that the framework can achieve promising object recognition and detection results that we report as a baseline.

1. INTRODUCTION

Interpretation of point cloud data is considered as an inevitable step in the development of a perceptual component of most of the recent robotic applications. It can provide useful information about the surrounding environment such as the location of objects and obstacles. Unlike image-based object detection, object detection based on point cloud can determine the exact 3D coordinate of the objects and help to plan the subsequent steps such as object manipulation or obstacle avoidance in a navigation scenario. Such a system can help to produce smart industrial robots for factories of the future, therefore scaling down the ergonomic concerns while improving the productivity and quality of the working environment.

Recent advancements in remote sensing technologies have been resulted in manufacturing sensors that capture 3D point clouds with a higher accuracy which makes them relatively robust against challenging scene characteristics (illumination, noise, etc.). In contrast, point clouds have irregular representations that make them not a suitable input for typical CNNs. To avoid the problem of irregularity in the point clouds, most of the current object detection approaches rely on methods based on 2D detectors. Such methods either extend 2D RGB detectors from images to detect objects in 3D or generate 2D images from point clouds in order to feed them to the detection network. For instance, in (Song, Xiao, 2016) and (Hou et al., 2019), an extended version of Faster and Mask R-CNN are applied to 3D. Usually, 3D irregular point clouds are voxelized and converted to regular grids, where a 3D detector is applied to those grids. These methods are usually subjected to high computational costs caused by costly 3D convolutions.

On the other hand, methods such as (Chen et al., 2017) (Zhou, Tuzel, 2018) project 3D point clouds to 2D bird's eye view (BEV) images, and similar to regular 2D cases, apply 2D detectors on the images. Despite being useful in some scenarios, these methods overlook enormous amounts of geometric information that can be critical, particularly in indoor environments. In such scenarios, as there are lots of clutter and occluded objects, using bird's eye view is not an effective option. For instance, in Frustum PointNet (Qi et al., 2018), there is a two-step detection network, such that 3D detections in the second step completely rely on the 2D detections in the first step. Using front-view RGB images, in the first step, objects are localized and their 2D bounding boxes are retrieved. In the second step, 3D frustums produced from the 2D boxes are utilized to localize objects in point clouds. These methods are highly dependent on 2D detectors in such a way that if the 2D detector can not detect an object, the 3D detector will also miss the object entirely in the point cloud.

Some architectures (Qi et al., 2017b) (Wang, Posner, 2015) utilize the sparsity of the cloud, only by considering the sensed points and convert them into regular structures. Directly processing the points can also prevent information loss caused by the quantization procedures. Therefore, a straightforward way for object detection is to represent the whole point cloud with such architectures (similar to the 2D detectors) and produce object proposals directly from the learned features. These approaches work in a two-dimensional case since the object center is a visible pixel in the image. However, this is not an advantageous solution in presence of sparsity, as is the case in 3D point clouds. In point clouds, the object center is not visible and it is usually far from the captured surface points. Such networks have therefore difficulty to aggregate the features in the local neighborhood of the object centers (unlike 2D cases). Increasing detection range can not even help, as it adds on further clutters and points from adjacent objects to the detection result. To mitigate this issue, some methods such as Hough VoteNet (Qi et al., 2019) propose a voting mechanism that generates points to estimate the object centers which are later will be aggregated to produce object proposals. This method is efficient and well-adapted for indoor scenarios, where there are lots of occlusions and methods based on bird's eye view will fail. Motivated from these studies, in this work, we

* Corresponding author

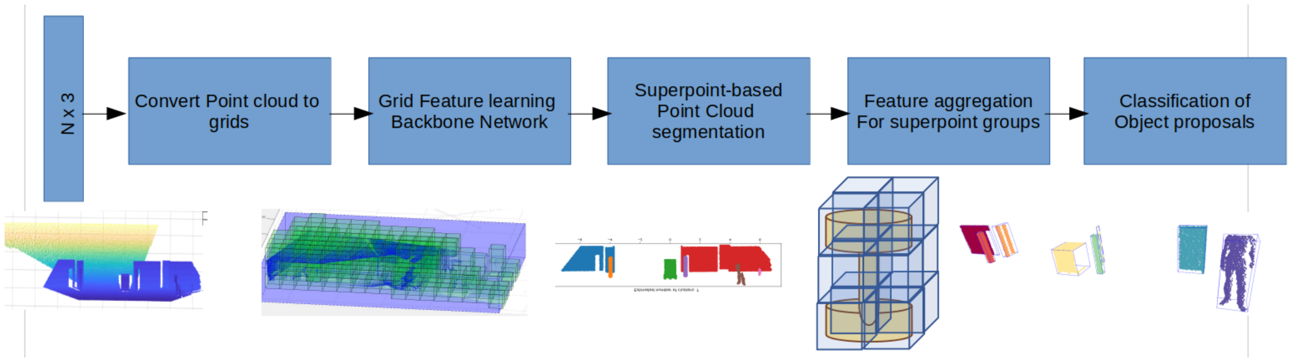
N x 3
Convert Point cloud to grids
Grid Feature learning Backbone Network
Superpoint-based Point Cloud segmentation
Feature aggregation For superpoint groups
Classification of Object proposals

Figure 1. Architecture of the proposed method illustrating object detection procedure

propose a new framework for object detection based on feature aggregation in the context of the factory of the future. The framework is adapted to improve the perceptual capabilities of Automated Ground Vehicles (AGVs) in the factory field. To this end, first, the acquired point cloud is passed to a preprocessing module that prunes the point cloud and generates the features. Then the point cloud is segmented into local patches, where features are aggregated and constitute the final object proposals. An object classifier iteratively trained through hard negative mining is used to determine object classes. Our contribution in this paper is three-fold: Firstly, we propose a new framework (Section 3) that is capable of object detection by feature aggregation from large point clouds using only point cloud information in cluttered indoor scenes. Secondly, we provide a challenging, partially-labeled synthetic object classification and detection dataset (Section 4) suitable for testing object detection frameworks. Our dataset introduces new challenges, specific for indoor industrial environments. Finally, using our developed framework, we produced baseline object recognition and detection results on collected dataset (Section 5). In addition, we evaluate its performance on challenging SUN RGB-D dataset using only geometric point clouds as input. We provide a detailed comparison to available methods in the literature that use only geometry or both geometry and RGB information.

2. RELATED WORK

2.1 3D Object Detection using Point Cloud

Previously, the proposed object detection methods were mostly as an extension of 2D approaches. Some methods use template-based detection approaches (Nan et al., 2012) (Li et al., 2015) and some others extended the use of sliding window detection (Song, Xiao, 2014) or feature engineering (Ren, Sudderth, 2016) to 3D. Object detection becomes challenging especially in terms of instance-based segmentation where most of the studies (Gupta et al., 2010) (Reitberger et al., 2009) (Wu et al., 2013) applied in urban environment in order to detect bounding boxes of individual objects in a cluster of similar objects, such as trees alongside road corridors.

Recently, there is a burst of interest in using deep neural networks that resulted a torrent of studies which achieved state-of-the-art performance in object detection. These networks perform efficiently when they work directly with 3D sparse inputs. In (Song, Xiao, 2016), they divided the point clouds into voxels and applied 3D CNN to learn features and to generate 3D bounding boxes. Another voxel-based method is VoxelNet (Zhou, Tuzel, 2018), which is an end-to-end deep network. It divides a point cloud into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation also known as Voxel Feature Encoding (VFE). Encoded point cloud is given to the region proposal network (RPN) to produce detections. PointPillars (Lang et al., 2019) is also an end-to-end architecture with 2D convolutional layers. It uses a novel encoder that learn features from pillars (vertical columns of the point cloud). First, the raw point cloud is converted into a stacked pillar tensors and pillar index tensors. The encoder uses the stacked pillars to learn a set of features that can be scattered back to a 2D pseudo-image for a convolutional neural network. The features from the backbone are used by detection head to predict 3D bounding boxes for objects. PointPillars run in 62 Hz and currently is the fastest method available. Transform, rotate and scale of raw point clouds are easy and straightforward. SECOND (Yan et al., 2018) uses this property to perform data augmentation on point cloud which boosts the performance of the network and speeds up the convergence process. It also introduces a new angular loss function that resolves the large loss values produced when the angular difference of the predicted bounding boxes and the ground truth bounding box are equal to π. The framework takes raw point cloud as input and converts it through a two layers voxel feature encoding (VFE) and a linear layer. Finally, a region proposal network produces the detections.

By increasing the complexity of working directly with 3D point cloud data, especially in an environment with a large number of points, some other types of methods (Ku et al., 2018, Ren et al., 2018) try to use projection to reduce the complexity. A popular approach in such methods is first to projects the 3D data into a bird's eye view before moving the data forward in the pipeline. They exploit the sparsity in the activation of CNNs to speed up inference. Similarly, ComplexYOLO (Simon et al., 2018) converts a point cloud into RGB BEV map, where the RGB map is encoded by a grid map using the CNN network. Inspired by the YOLO architecture for 2D object detection (Redmon et al., 2016), the network predicts five boxes per grid cell. Each box prediction is composed by the regression parameters and object scores with a general probability and class scores.

However, the bird's eye view projection and voxelization and in general 2D image-based proposal generation suffers from information loss due to the data quantization. These methods might fail in indoor challenging scenarios because such scenes can only be observed from 3D space and there is no way for a 3D object box estimation algorithm to recover these kinds of failures.

2.2 Learning Point Cloud Representations for Detection and Segmentation

Instead of the above-mentioned methods that represent the point clouds as voxels or in multimodal formats, these methods learn features directly from the raw point cloud. This direct utilization of point cloud is extremely efficient and increases the speed of the classification and segmentation architectures. PointNet (Qi et al., 2017a) learns a higher dimensional spatial feature representation for each 3D point and then aggregates all the points within a small 3D volume (typically an occupancy grid cell) in order to model a 3D neighborhood context. However, PointNet lacks the ability to capture local context at different scales. The follow-up work PointNet++ (Qi et al., 2017b) introduced a hierarchical feature learning framework that improved the feature extraction quality by considering the local neighborhood of the points. Direct interaction of these methods with the raw point data makes them efficiently run in real-time.

Most of these methods work on outdoor autonomous driving scenarios to detect cars and pedestrians and use BEV for representing the point clouds. Therefore, they are not a suitable choice for an indoor scenario, where there are lots of occlusions and clutter in the scene. In a factory, there are lots of racks where lots of objects will be occluded and therefore BEV representation will end up an enormous loss of 3D information, which accordingly will lead to poor detection results. Plus, a good many methods utilize RGB information as input. Although images are rich in perceptual information, our proposed framework considers efficiency and confidentiality in industrial factories by using only point cloud information. Moreover, it is flexible in using various feature extraction backbones. The feature aggregation module can use any kind of feature that is provided by the feature extraction module of the framework.

3. PROPOSED FRAMEWORK

The proposed framework consists of five main components (see Figure 1): a preprocessing block that takes a point cloud as input and performs several operations such as pruning, cropping, calculating grids and producing sparse representation. The resulted voxels are used for calculating features which represent the geometry of the contained points by a feature extraction network. Based on the calculated features, a segmentation method is used for partitioning the patches of the calculated local features into individual object segments. The aggregation block takes these segments and calculates the final object proposal feature by concatenating features of the constituent voxels. Finally, a deep convolutional network based classifier identifies the class type of the proposal. The classifier is initially trained with a set of labeled object instances as well as a small set of a background class. The classifier makes two types of errors: false positives and false negatives. After several iterations, the false positives are incrementally collected in the background set. Therefore, the classifier is iteratively trained with on-the-fly generated instances.

3.1 Pre-processing

The input for the ground surface detection algorithm is a translated point cloud D from each scan. The point cloud is translated in a way that the ego-vehicle is the origin of the coordinate system. Therefore, P_i can be considered as a set of the 3D point clouds at time i and $P = \{P_{i-m}, \dots, P_{i-1}, P_i\}$ as the set of current and last m point clouds. Similarly, set $N = \{N_{i-m}, \dots, N_{i-1}, N_i\}$ can be assumed as translation and rotation parameters of the vehicle in the Euclidean space, where each consists of a three by three rotation and a one by three translation vector: $N_i = [R_i|T_i]$. To transform the point cloud in the current time from vehicle coordinates to global coordinates, we can apply the following transformation to the point cloud P: $R_i \times P_i + T_i$. Moreover, we can combine several scans to create a dense point cloud, since 12 sensors are available in the recorded dataset. When the transformation parameters among the sensors are available, we can merge points from different sensors by aligning the source cloud to the target point cloud coordinate to construct the dense point cloud. This process can be done efficiently by the Iterative Closest Point (ICP) algorithm (Rusinkiewicz, Levoy, 2001). The output is a point set of tightly aligned point clouds. Finally, a combined dense point cloud is obtained knowing the correct transformation from ICP ($N_c == [R_c|T_c]$) to our target reference frame (ego-vehicle) and the parameters of each scan N_i:

$$D_i = \cup R_k^{-1} \times ((R_c \times P_c + T_c) - T_k) \tag{1}$$

where $\cup$ combines the points from each $k = 1, \dots, K$ sensor. The crop operator ($Crop(D_i)$) is also introduced that can crop the dense point cloud in a range from the ego vehicle. This way, a range value can be assigned by considering different criteria. For instance, a range very close to the vehicle can be considered as the critical region and object detection algorithm can be used in this cropped region with higher accuracy. A pruning operator ($Prune(D_i, r)$) is utilized that takes the given point cloud and prunes it in a given range. This way, we sub-sample the point cloud to a lower number of points to gain efficiency. By default, we prune the point clouds at $30\,\mathrm{cm}$. Finally, the point cloud is discretized into fixed-dimensional grid voxels. Compared to RGB images, 3D point clouds are sparse and most of the space in such data is unoccupied. To exploit the sparsity of the point cloud only occupied grids are kept and the rest is discarded. This is critical for feature extraction step as it maps unoccupied grids to zero feature vector.

3.2 Feature Extraction

We refer to feature vector of a grid G at location (i, j, k) by $f_{i,j,k}$. If we consider that each grid has a dimension of (N_x, N_y, N_z), then we can define set Φ as the set of grid indices in the feature grid to keep track of occupied and unoccupied grid cells. Therefore, $\phi \in \Phi$ indicates a specific set of voxel indices in the whole grid. If the voxel is unoccupied, then its feature set is mapped to zero ($f_\phi = 0$) in order to help to generate a sparse feature set. Our framework is flexible and can utilize any feature extraction head for the occupied voxels. In this work, we use a modified version of (Ben-Shabat et al., 2017) for feature extraction. This object recognition network is adapted to extract features from a large point cloud grid with a given grid dimension and create its sparse grid set ϕ. This set is used for feature aggregation in later steps. The utilized network generalizes the Fisher Vector representation (Sánchez et al., 2013) to describe 3D point cloud data uniformly in order to use them as input for deep convolutional architectures. Therefore, to extract features of only one voxel, if $X = p_t$ where $t = 1 \dots T$ is considered as a set of T 3D points in voxel (G_i of ϕ) of a given point cloud, we can define a mixture of Gaussian with k components and determine the likelihood of a single point p_t of X associated with k^{th} component as:

$$u_k(p) = \frac{1}{2\pi^{D/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(p-\mu_k)' \sum_k^{-1}(p-\mu_k)} \tag{2}$$

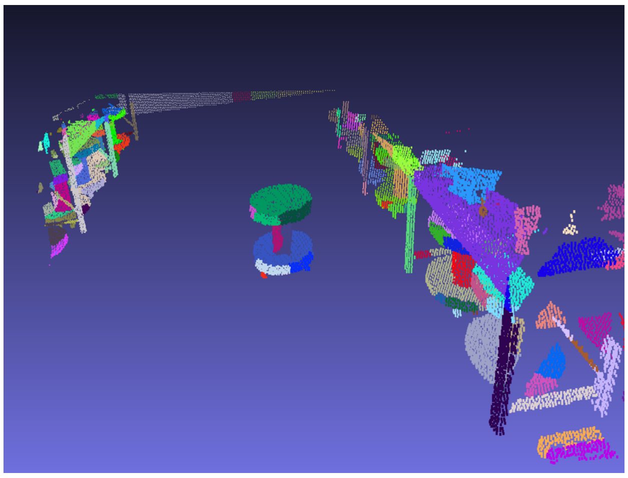

Figure 2. Result of over-segmentation for generating local object patches before applying min-cut optimization to final object segmentation

where D is the number of dimensions of the data points. μ and $\sum$ are the mean and covariance, respectively. Then, we can calculate likelihood of all points belonging to G_i to mixture of all k Gaussian components as:

$$u_G(X) = \sum_{k=1}^{K} w_k u_k(X) \tag{3}$$

with w_k the weight of k^{th} Gaussian component. Therefore, given a specific GMM with parameters λ and point set X we can normally calculate its Fisher vector $\mathscr{G}_\lambda^X$ which is sum of normalized gradients:

$$\mathscr{G}_\lambda^X = \sum \nabla_\lambda \log u_\lambda(p_t) \tag{4}$$

the soft assignment of each point in the voxel to the GMM components can be calculated with its normalized gradient components ($\mathscr{G}_{wk}^X$, $\mathscr{G}_{\mu k}^X$,$\mathscr{G}_{\sigma k}^X$) and get concatenated to produce the final Fisher vector representation of the voxel:

$$\mathscr{G}_{G_i}^X = (\mathscr{G}_{\sigma_1}^X \ldots \mathscr{G}_{\sigma_k}^X, \mathscr{G}_{\mu_1}^X \ldots \mathscr{G}_{\mu_k}^X, \mathscr{G}_{w_1}^X \ldots \mathscr{G}_{w_k}^X) \tag{5}$$

The feature vector is normalized to sample size. This is a hybrid representation that combines the discrete nature of the grids with the continuous structure of the Gaussian mixtures. The calculated features create unique representations for grids independent of the number of the points as well as invariant to both permutation and feature sizes.

3.3 Segmentation into Local Patches

For segmentation of the point cloud into individual objects, we follow $l_0 - cut$ greedy approach (Landrieu, Obozinski, 2016) which is a graph-cut method based on the min-cut algorithm (Kohli, Torr, 2005). It first over-segments a given point cloud into various partitions using its nearest neighborhood graph. Later, it iteratively calls min-cut algorithm to minimize the segmentation error. If we describe a point cloud's geometrical structure with an unoriented graph $G = (V, E)$ (nodes as points in the point cloud and edges as their adjacency relationship), we can achieve optimal segmentation error by optimizing a Potts segmentation energy function (g) and produces individual object segments:

$$\underset{g \in R}{\operatorname{argmin}} \sum_{i \in V} ||g_i - f_i||^2 + \rho \sum_{i,j \in E} \delta(g_i - g_j) \tag{6}$$

where f_i is acquired local geometric features. This formulation is beneficial as it uses a regularization parameter (ρ) that determines the number of clusters. There is also no need to set the maximum size of the segments and therefore objects in various sizes can be retrieved. By solving the optimization problem, the algorithm generates K non-overlapping partitions $S = (S_1, \ldots, S_k)$ which is used as input for the aggregation block of the framework. Figure 2 illustrates the process of generating local object segments.

3.4 Feature Aggregation

Given the feature sparsity set f_ϕ and the generated segments S, object proposal can be generated for the classification step. Set f indicates non-sparse grid cells that occupy each generated segment S_i. By retrieving these cells, the final representation of the proposals can be calculated by aggregation of the cell features that comprise the segments which are simply a normalized combination of the occupied cell's Fisher vector descriptors:

$$F = \frac{1}{n} \sum_{j=1}^{n} f_{\phi,j} \dot{w}_{\phi,j} \tag{7}$$

where n is the number of the segments, j is the non-occupied cell index and w is the weight of the features that can be determined based on the feature type or location of the cell in the grid.

3.5 Classification

The aggregated point cloud features are received by convolutional network architecture proposed in (Ben-Shabat et al., 2017) and the object models are trained through back propagation and optimization of the standard cross-entropy loss function using batch normalization. The training starts with a small initial set of background class instances and continues with standard hard negative mining technique inspired by common image-based object detectors (e.g. (Felzenszwalb et al., 2009)). In each iteration, the classifier is evaluated on the training set and all the false positive detections are stored. These negative samples then are listed in decreasing order and the first N samples are collected and added to the background class samples. Finally, the classifier is trained by the updated training instances and this process is repeated for several iterations until the desired result is achieved.

3.6 Post-processing

The detection process can result in multiple overlapping object bounding boxes because of the dense nature of the dataset that includes lots of objects placed in a close range from each other. To avoid this problem, we employ a 3D non-maximum suppression approach which has been widely used in many detection tasks (Neubeck, Van Gool, 2006, Felzenszwalb et al., 2009). First, the detection bounding boxes are listed with descending order of their detection scores. By comparing those on top of the list with the currently accepted object list, we can make a decision to keep them or not. If the overlap of the bounding box with the previously accepted one is no more than a given threshold then, it is kept. The bounding boxes overlap is calculated by intersection over union (IoU) metric.

Object category	Barrel	Bobbin	Box	Cone	Pallet	Person	Truck
No. of instances	1664	28961	32057	404	5651	888	113

Table 1. Details of the annotated objects in the synthetic dataset

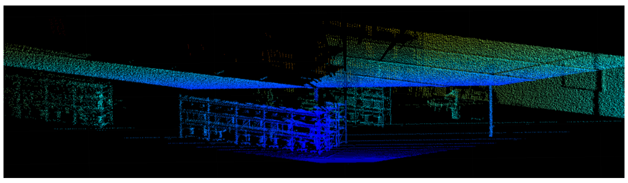
Figure 3. Shows a point cloud scan from the factory model in the synthetic dataset

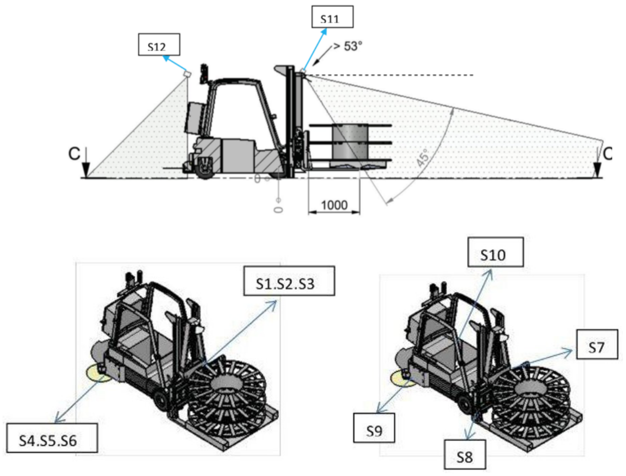

Figure 4. Vehicle and configuration of the twelve sensors mounted on it

4. DATASETS

4.1 Synthetic Dataset

We introduce novel simulated dataset representing a scenario in a modeled factory. In the simulation, a ground vehicle starts from a specific point in the 3D model of a factory and navigates throughout the factory and comes back to its primary position. Figure 3 shows a point cloud retrieved from a scan captured from the simulated 3D factory model. As it can be partially seen from the point cloud, there are lots of racks in the model where at each row of the rack various object types are located. The objects are interconnected with racks and also the other nearby object. This introduces difficulties for segmentation and makes object detection in this dataset a very challenging task.
As shown in Figure 4 twelve sensors ($S1$ to $S12$) are mounted on the vehicle. The 3D sensors have a 120° horizontal and 45° vertical field-of-view. Each sensor has its own yaw, pitch and raw angles and records with a maximum scan range of about 100 m.

We evaluate our framework on this dataset. The dataset contains 3 minutes and 20 seconds of simulation consisted of 6389 scans. The scans are recorded in 30 scans per second rate and each scan on average includes 80 000 points. For evaluations, we selected 2/3 of the scans (4260 frames) for training and the rest for testing. For object recognition/detection, there are around 344 objects annotated in the global map of the factory.

Considering the whole scans, the total number of the annotated object instances in the dataset are 69 738. The initial instances

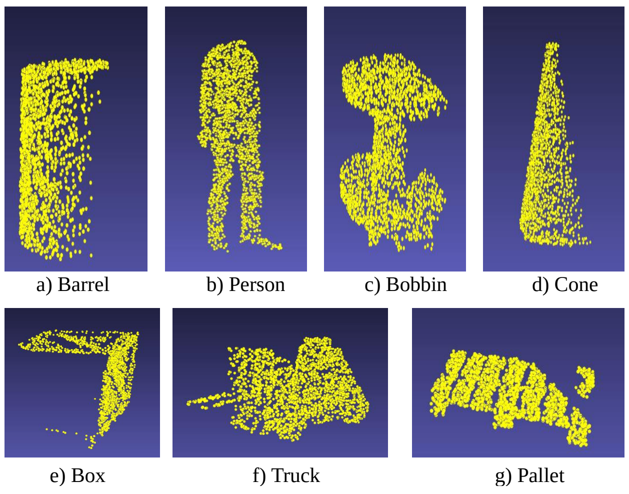

Figure 5. The simulated dataset from the factory model

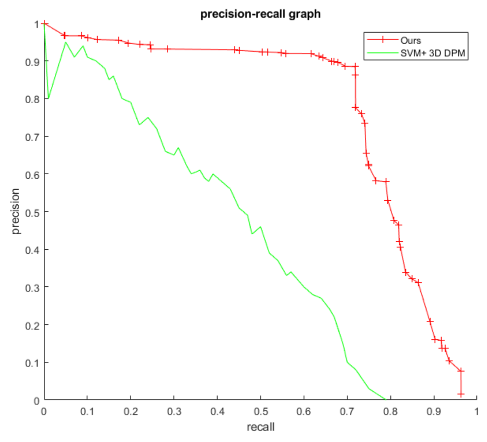

Figure 6. Precision-Recall curve of the whole dataset after iterative hard negative mining. The curve is obtained by setting different detection thresholds and is compared with SVM 3D-DPM baseline

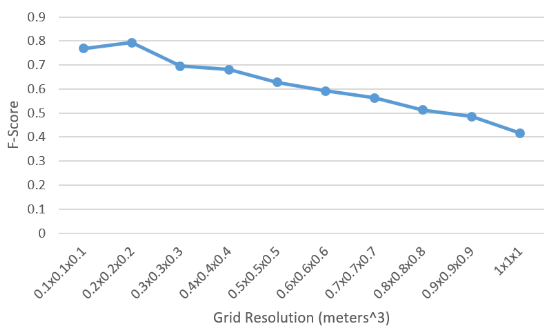

Figure 7. The effect of grid resolution parameter on accuracy of detections on the synthetic dataset

Object category	Barrel	Bobbin	Box	Cone	Pallet	Person	Truck	Total
Accuracy (%)	99.07	98.55	98.05	95.92	98.99	99.21	100	99.84

Table 2. Results of object recognition on the synthetic dataset using the trained deep classifier

	Barrel	Bobbin	Box	Cone	Pallet	Person	Truck	Total (F-Score)
SVM 3D-DPM (Pepik et al., 2012)	**0.12**	0.634	0.327	0.241	0.396	0.497	0.194	0.417
Ours	0.038	**0.964**	**0.698**	**0.457**	**0.651**	**0.773**	**0.258**	**0.793**

Table 3. Results of the object detection on the synthetic dataset using the proposed framework

in the background class are around 2000. With the background class, in total, there are 8 object classes in the dataset. The annotated objects consist of at least 2048 points. Every frame in the dataset is accompanied by an RGB image. Figure 5 depicts instances of the available objects in the dataset and Table 1 shows the details of the dataset. Unlike other recent datasets recorded for autonomous driving scenarios which main focus is on cars, pedestrians and bikes, our dataset includes seven different object classes emphasizing on industrial environment that are not addressed by most of the current solutions.

4.2 SUN RGB-D

SUN RGB-D is a scene understanding 3D benchmark dataset consisting of 37 object classes. The dataset includes around 10K RGB-D indoor images with accurately annotated object categories and orientations. Prior to feeding the dataset into our framework, the RGB-D images are converted to point cloud data. For evaluation, we followed the standard protocol provided in the original study and for comparison, we use 10 commonly used object categories. This dataset is used to evaluate performance of the proposed framework in real-world scenarios, enabling us to compare with state-of-the-art.

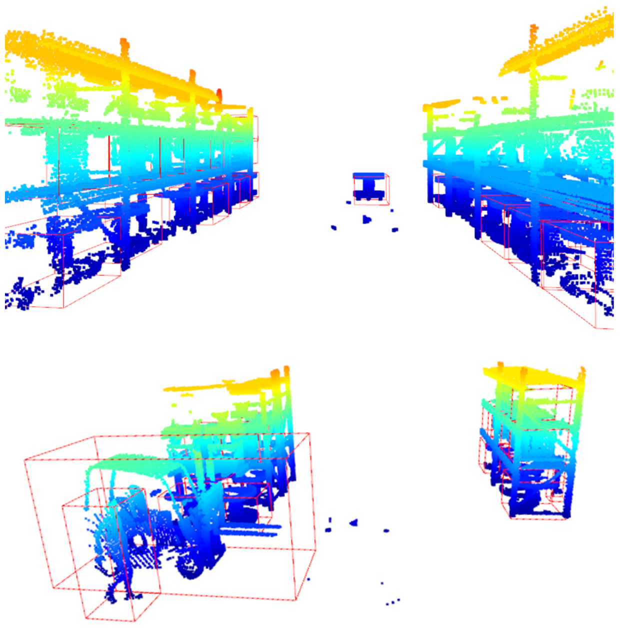
Figure 8. Qualitative results of the proposed object detection framework applied on a sample scene from our synthetic dataset

5. EXPERIMENTS

To evaluate performance of the framework, we use mean average precision (mAP) and F-score metrics (calculated as $F-score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$). The intersection over union threshold for the reported best results is set to 0.6 for the synthetic dataset while it is set to 0.25 for SUN RGB-D dataset. Table 2 shows the results of the object recognition using the trained final classifier. For training the network, the batch size is set to 64, decay rate to 0.7 and learning rate to 0.001. For the representation of the feature vectors, the number of Gaussian functions and their variances are empirically chosen to be 5 and 0.04, respectively. The minimum number of points for an object is 2048. For optimization, Adam optimizer is used. For training, only 50 epochs were used on an Nvidia Geforce RTX 2060 and each training iteration took about two hours on synthetic (and 10 hours on SUN RGB-D datasets). The obtained average class accuracy is 99.84**%** (mean loss is 0.010). As it can be noticed, the trained classifier achieved a very high recognition accuracy. It is even possible to further increase accuracy by increasing the number of epochs at each iteration. Table 3 and Figure 8 show object detection results on our dataset. The table presents the results which are obtained after the final training iteration (the hard negative mining iteration is set to 50). We achieve satisfactory performance in most of the classes. However, for some classes, such as “Barrel” and “Truck”, the detection results are not desirable. This is related to several problems. In some scenes (especially, the ones including “Barrels”) the object points are so interconnected that the segmentation algorithm fails to separate the object and therefore, generates improper proposals. Moreover, the provided dataset is partially annotated. Thus, for some classes, such as “Box” and “Bobbin”, the classifier performs a correct detection of the objects (TP) which are not initially annotated in the ground-truth (“Bobbin” instances on the back row of the racks on the right side of Figure 8). Those instances can be added to the background class in the training set and reduce the accuracy of the classifier and detection. The dataset is also not balanced and for some classes, such as “Cone” and “Truck”, the training data is not sufficient. The Precision-Recall curve in Figure 6 underlines that despite all these challenges, the overall accuracy of the baseline framework is promising. Resolution of the grid cell is an important factor in producing accurate detections. Figure 7 depicts influence of this parameter on performance. As the resolution of grids increases, the performance increases until it tops when the grid resolution is 0.2 m^3. Increasing grid resolution further makes the feature blocks bigger hence less detailed. Accordingly, the feature aggregation is affected in a negative way which decreases the performance. We compare performance of our framework on synthetic dataset with (Pepik et al., 2012) which is an extension of 2D deformable part models to 3D. Table 3 and Figure 6 show that our framework significantly outperforms this baseline (by 0.39 in F-score). Higher performance of 3D-DPM method (which is not a CNN based method) on the “Barrel” category explains the low performance of our method on classes with low number of instances and thereby

Method	Input type	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
DSS (Song, Xiao, 2016)	Geo & RGB	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	42.1
COG (Ren, Sudderth, 2016)	Geo & RGB	58.3	63.7	31.8	62.2	**45.2**	15.5	27.4	51.0	**51.3**	70.1	47.6
2D-driven (Lahoud, Ghanem, 2017)	Geo & RGB	43.5	64.5	31.4	48.3	27.9	25.9	41.9	50.4	37.0	80.4	45.1
Ours	Geo only	61.3	**84.9**	29.2	44.2	37.5	26.4	52.7	62.4	47.2	88.4	53.4
F-PointNet (Qi et al., 2018)	Geo & RGB	43.3	81.1	**33.3**	64.2	24.7	**32.0**	58.1	61.1	51.1	**90.9**	54.0
VoteNet (Qi et al., 2019)	Geo only	**74.4**	83.0	28.8	**75.3**	22.0	29.8	**62.2**	**64.0**	47.3	90.1	**57.7**

Table 4. Results of 3D object detection on the SUN RGB-D validation set

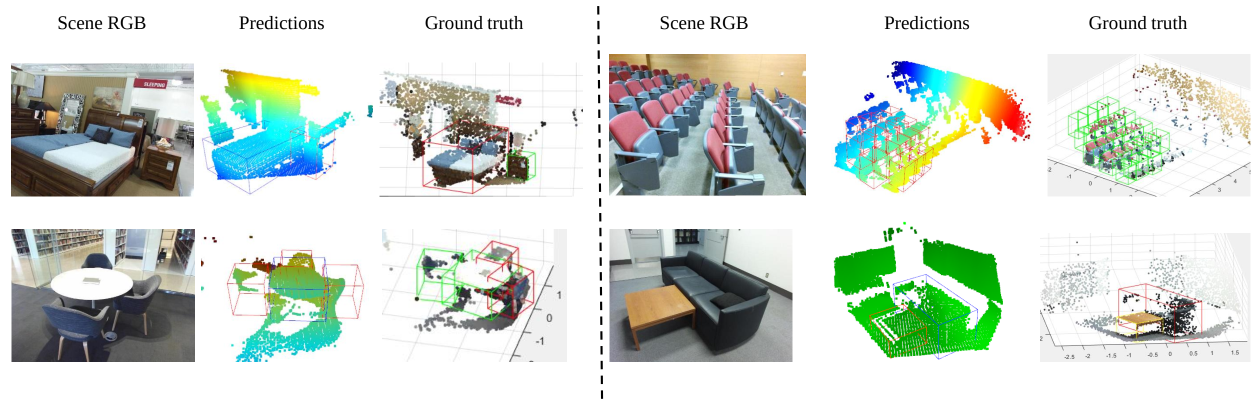

Figure 9. Examples of qualitative detection results on SUN RGB-D dataset. In all set of images from left to right: a RGB image from test set, predicted 3D bounding boxes by our framework and annotated ground-truth

the network has difficulty to converge in those categories.

We also made a comparison to prior state-of-the-art methods using SUN RGB-D dataset. To be able to compare with these methods, we use mean Average Precision (mAP) metric. Notice that we use the same set of hyperparameters to train object detection network in both datasets. Deep Sliding Shapes (DSS) (Song, Xiao, 2016) is a 3D CNN based method that uses RGB images with 3D information together. Cloud of Oriented Gradients (COG) (Ren, Sudderth, 2016) is a sliding window based detection method that produces HOG like descriptors for 3D detectors. 2D-driven (Lahoud, Ghanem, 2017) and F-PointNet (Qi et al., 2018) use 2D detectors to reduce search space for 3D detection. VoteNet (Qi et al., 2019) is current state-of-the-art method that uses Hough voting for 3D object detection using only point cloud information. As shown in Table 4, we achieve competitive results compared to the current best methods in the literature. Out of ten categories, we achieve the best performance in one category ("bed") and second best performance on three other categories. It achieves third best overall performance, however, notice that our method and VoteNet are the two methods that use only geometric information. Other methods use both RGB and geometry information to perform object detection. Figure 9 represents qualitative examples of our framework on SUN RGB-D dataset. Despite various challenges introduced in these scenes (e.g. cluttered, attached objects etc.), our framework produces robust detection bounding boxes. For example, in bottom right and bottom left of the Figure 9, the chairs and the sofa are attached to the tables, however, the framework is able to distinguish between them and produce correct detection boxes. It is interesting to see that interconnected objects make more trouble for the network in synthetic dataset as the density of the point clouds is much higher than a real scene. It therefore fails to produce correct proposals and misses on those objects. Similar to synthetic dataset that the framework was able to detect several similar attached objects in a scene (such as "Bobbin"s and "Boxes"), it also detects vast majority of similar objects such as chairs in real scenes (top right of Figure 9). Like other methods, the proposed framework has some limitations. Difficulties in segmentation of very dense and attached objects and false hallucinations of occluded objects are among them. The future work will target these issues to further increase the performance in object detection task.

6. CONCLUSION

In this paper, a novel object detection framework is proposed with a focus on the indoor factory environment. Unlike most of the recent state-of-the-art methods that use image and point cloud together, it only uses point cloud information. The features of the grid representation are computed and stored as a sparse representation. These features are restored and aggregated after local segmentation of the objects. A deep classifier trained with generalized Fisher representation is employed to learn object models. The developed framework is evaluated on a dataset which simulates a factory of the future setting and also SUN RGB-D object detection benchmark dataset. Based on the evaluations, the proposed framework achieves a competitive performance in object recognition as well as object detection task.

ACKNOWLEDGEMENTS

This work was sponsored by a public grant overseen by the French National Research Agency as part of the "Investissements d'Avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-0016) and the IDEX-ISITE initiative CAP 20-25 (ANR-16-IDEX-0001). Financial support was also received from the European Union through the European Regional Development Fund program (ERDF – AURA region) and by the Auvergne-Rhône-Alpes region.

REFERENCES

Ben-Shabat, Y., Lindenbaum, M., Fischer, A., 2017. 3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks. *arXiv preprint arXiv:1711.08241*.

Chen, X., Ma, H., Wan, J., Li, B., Xia, T., 2017. Multi-view 3d object detection network for autonomous driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1907–1915.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., Ramanan, D., 2009. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9), 1627–1645.

Gupta, S., Weinacker, H., Koch, B., 2010. Comparative analysis of clustering-based approaches for 3-D single tree detection using airborne fullwave lidar data. *Remote Sensing*, 2(4), 968–989.

Hou, J., Dai, A., Nießner, M., 2019. 3d-sis: 3d semantic instance segmentation of rgb-d scans. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4421–4430.

Kohli, P., Torr, P. H., 2005. Efficiently solving dynamic markov random fields using graph cuts. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2, IEEE, 922–929.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S. L., 2018. Joint 3d proposal generation and object detection from view aggregation. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 1–8.

Lahoud, J., Ghanem, B., 2017. 2d-driven 3d object detection in rgb-d images. *Proceedings of the IEEE International Conference on Computer Vision*, 4622–4630.

Landrieu, L., Obozinski, G., 2016. Cut Pursuit: fast algorithms to learn piecewise constant functions. *Artificial Intelligence and Statistics*, 1384–1393.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O., 2019. PointPillars: Fast encoders for object detection from point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12697–12705.

Li, Y., Dai, A., Guibas, L., Nießner, M., 2015. Database-assisted object retrieval for real-time 3d reconstruction. *Computer Graphics Forum*, 34(2), Wiley Online Library, 435–446.

Nan, L., Xie, K., Sharf, A., 2012. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6), 1–10.

Neubeck, A., Van Gool, L., 2006. Efficient non-maximum suppression. *18th International Conference on Pattern Recognition (ICPR'06)*, 3, IEEE, 850–855.

Pepik, B., Gehler, P., Stark, M., Schiele, B., 2012. 3d 2 pm–3d deformable part models. *European Conference on Computer Vision*, Springer, 356–370.

Qi, C. R., Litany, O., He, K., Guibas, L. J., 2019. Deep Hough Voting for 3D Object Detection in Point Clouds. *arXiv preprint arXiv:1904.09664*.

Qi, C. R., Liu, W., Wu, C., Su, H., Guibas, L. J., 2018. Frustum pointnets for 3d object detection from rgb-d data. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 918–927.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 5099–5108.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Reitberger, J., Schnörr, C., Krzystek, P., Stilla, U., 2009. 3D segmentation of single trees exploiting full waveform LIDAR data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6), 561–574.

Ren, M., Pokrovsky, A., Yang, B., Urtasun, R., 2018. Sbnet: Sparse blocks network for fast inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8711–8720.

Ren, Z., Sudderth, E. B., 2016. Three-dimensional object detection and layout prediction using clouds of oriented gradients. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1525–1533.

Rusinkiewicz, S., Levoy, M., 2001. Efficient variants of the icp algorithm. *3dim*, 1, 145–152.

Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J., 2013. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3), 222–245.

Simon, M., Milz, S., Amende, K., Gross, H.-M., 2018. Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds. *European Conference on Computer Vision*, Springer, 197–209.

Song, S., Xiao, J., 2014. Sliding shapes for 3d object detection in depth images. *European conference on computer vision*, Springer, 634–651.

Song, S., Xiao, J., 2016. Deep sliding shapes for amodal 3d object detection in rgb-d images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 808–816.

Wang, D. Z., Posner, I., 2015. Voting for Voting in Online Point Cloud Object Detection. *Robotics: Science and Systems*, 1(3), 10–15607.

Wu, B., Yu, B., Yue, W., Shu, S., Tan, W., Hu, C., Huang, Y., Wu, J., Liu, H., 2013. A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data. *Remote Sensing*, 5(2), 584–611.

Yan, Y., Mao, Y., Li, B., 2018. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 3337.

Zhou, Y., Tuzel, O., 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.