Online Point Cloud Object Recognition System using Local Descriptors for Real-time Applications

Yacine Yaddaden¹,², Sylvie Daniel¹,³ and Denis Laurendeau¹,²

¹Laboratoire de Vision et Systèmes Numériques (LVSN), Université de Laval, ville de Québec, Québec, Canada
²Département de Génie Électrique et de Génie Informatique, Université de Laval, ville de Québec, Québec, Canada
³Département des Sciences Géomatiques, Université de Laval,

Keywords: Online Machine Learning, Point Cloud Data, Unique Shape Context, Signature of Histograms of Orientation, Principal Component Analysis, Multi-class Support Vector Machine.

Abstract: In the context of vehicle localization based on point cloud data collected using LiDAR sensors, several 3D descriptors might be employed to highlight the relevant information about the vehicle’s environment. However, it is still a challenging task to assess which one is the more suitable with respect to the constraint of real-time processing. In this paper, we propose a system based on classical machine learning techniques and performing recognition from point cloud data after applying several preprocessing steps. We compare the performance of two distinct state-of-the-art local 3D descriptors namely Unique Shape Context and Signature of Histograms of Orientation when combined with online learning algorithms. The proposed system also includes two distinct modes namely normal and cluster to deal with the point cloud data size and for which performances are evaluated. In order to measure the performance of the proposed system, we used a benchmark RGB-D object dataset from which we randomly selected three stratified subsets. The obtained results are promising and suggesting further experimentation involving real data collected from LiDAR sensors on vehicles.

1 INTRODUCTION

During the last decades, we have witnessed the emergence of new acquisition devices and sensors which allow to capture more reliable and relevant information. Indeed, new camera models (such as the Microsoft Kinect) and wide range sensors are employed to capture depth information. Moreover, these sensors are becoming more affordable which leads to create new fields of application. One of the most interesting use of this kind of sensors remains autonomous vehicle or self-driving car (Lee et al., 2016). Several systems have been proposed aiming to exploit the potential of depth information in order to avoid obstacles or for localization (Wolcott and Eustice, 2014).

The main aim of this work lies in developing an efficient recognition system from point cloud data that will be used in the context of driving assistance and, more precisely, for localization in real-time. In this paper, we present a comparison in terms of performance of the proposed system when using different techniques in the processing blocks. Indeed, we compare two distinct local 3D descriptors namely USC and SHOT (including the color version SHOT-rbg) for data representation. We also compare two operating modes, namely normal and cluster, in order to deal with variable size of the point cloud. However, the most important improvement in comparison to traditional systems lies in the use of incremental or online versions of the learning algorithms. They allow to tackle the memory issue caused by processing huge amounts of data. In order to evaluate the performance of the proposed system, we use a common and publicly available benchmark RGB-D dataset (Lai et al., 2011).

The remainder of the paper is structured as follows. In section 2, we present basic notions about recognition systems from point cloud data. Section 3 details each component of the proposed system. Sections 4 and 5 are dedicated to the description of the experimental protocol for system evaluation with several metrics and the discussion of the results. Finally,
in section 6 concluding remarks and further works are presented.

2 BACKGROUND

2.1 Fundamentals

Object recognition systems from point cloud data consist of the same building blocks as those of common pattern recognition system. Indeed, they include an input which is fed by raw point cloud data and an output which represents the label of the recognized object. Between them, there are several processing blocks, some of which are mandatory and other optional. The preprocessing block aims to prepare and clean up the input point cloud data before the feature extraction step. This building block is critical since it aims to generate an efficient and robust representation using either handcrafted descriptors (Guo et al., 2016) or new deep learning techniques (Zaki et al., 2016) and (Schwarz et al., 2015). In order to reduce the size of the generated representation to optimize the computation time, dimension reduction is employed to highlight discriminant information while getting rid of the redundant one. Finally, the classification block exploits a supervised machine learning technique to perform recognition (Maturana and Scherer, 2015).

Working with point cloud data and depth information is not new. In fact, several 3D descriptors have already been proposed in order to provide a reliable representation from raw data. Moreover, numerous works have been conducted in order to provide an accurate comparison in terms of performance of state-of-the-art and most common 3D descriptors (Guo et al., 2016), (Hana et al., 2018) and (Carvalho and von Wangenheim, 2019). We distinguish three different categories of descriptors: 1) local descriptors which encode the local geometric information at each point based on its neighbors, 2) global features (do Monte Lima and Teichrieb, 2016) that highlight the geometric information of the whole 3D point cloud, 3) hybrid descriptors (Alhamzi et al., 2015) which combine and group the essential information provided by both previous representations in the sake of performance improvement. Each one has its strengths and limitations. Indeed, local descriptors are more robust and less sensitive to partial occlusion, but remain computationally inefficient in comparison to global ones. The computational inefficiency of local descriptors lies in the huge size of the extracted representation. However, it is possible to get rid of this flaw by reducing the size of the generated representation by applying dimension reduction.

It is important to highlight the fact that depending on the field of application, the point cloud size might vary. Indeed, when using point cloud data collected with LiDAR sensors in the context of localization, the amount of data is too large to be processed in one shot. Usually, in order to deal with this type of situation, the point cloud is partitioned using dedicated techniques, then each segment is processed individually. However, in the case of small object recognition, the partitioning step is not required.

As mentioned above, several descriptors can be used in order to generate a relevant representation from point cloud data and choosing the most adequate one is tough since it depends on the context. In our case, details are important and contribute to improve the discrimination capability of the developed system when performing localization. Thus, the choice of local 3D descriptors seems adequate. The hybrid ones might also do the job, but in the context of real-time application, reducing the amount of computation is mandatory and combining two descriptors does not help. Among all the existing local 3D descriptors, we chose USC for its capability to decrease memory requirement while improving accuracy (Guo et al., 2014). As for our primary choice, we chose SHOT and its color version SHOT-rgb since it is highly descriptive, computationally efficient and robust to noise (Guo et al., 2014), (Alexandre, 2012) and (Hana et al., 2018).

USC stands for Unique Shape Context and consists in a local 3D descriptor proposed by (Tombari et al., 2010). USC is an improvement and optimization of the 3DSC (Frome et al., 2004) aiming to reduce the computational load. In order to generate this representation, the first step consists in constructing a LRF (Local Reference Frame) before aligning the neighboring points to ensure invariance to translations and rotations. As shown in Figure 1a, the neighborhood of the keypoint from which is extracted the descriptor is divided into several bins. The final histogram is computed by accumulating the weighted sum of the points in each bin.

SHOT or Signature of Histograms of Orientation is also a local 3D descriptor introduced by (Salti et al., 2014). It is one of the most effective 3D features in terms of descriptiveness and computational efficiency (Hana et al., 2018). Just like the USC descriptor, the first step consists in the construction of a LRF and aligning the neighboring points. As shown in Figure 1b, the sphere around the keypoint is divided into several volumes along the azimuth, radial and elevation axes. Then, for each volume, a local histogram is
computed by counting the points into bins according to the angles between the normals at the neighboring points inside the volume and the normal at the key-point. The last step consists in concatenating all the local histograms.

One of the common issues when working with point cloud data lies in the huge number of points to process. In order to deal with this situation, keypoints selection has to be included in the system’s building blocks in order to reduce the size of the input data and at the same time improving computer efficiency. Basically, it consists in selecting a certain amount of points from the initial point cloud using a specific technique. Then, instead of generating the representation from a large point cloud, the operation is performed on a smaller one. Several techniques exist and might be employed, (Filipe and Alexandre, 2014) presented an accurate comparison in terms of performance. Among the different existing techniques we find: Harris3D, SUSAN, SIFT3D and ISS3D, the best performance being achieved by the two latter techniques. However, the presented list is not exhaustive and several other techniques might be used.

2.2 Related Works

As discussed previously, there are several building blocks for object recognition systems applied to point cloud data. Besides, we distinguish two categories of pipelines. The traditional one is based on handcrafted representation and classic learning techniques for recognition. (Chen et al., 2018) have employed several local 3D descriptors, among which figure the USC and SHOT, in order to perform construction object recognition. Moreover, they used matching learning techniques for recognition. (Cop et al., 2018) proposed a new system called DELIGHT that allows a robot to estimate its position based on LiDAR data and using the SHOT descriptor. Similarly, (Guo et al., 2019) proposed a localization system in the context of mobile robotics using a new local 3D descriptor ISHOT (Intensity SHOT). It yields the best performances when compared with other 3D descriptors. (Garstka. and Peters., 2016) have compared the performance of several local 3D descriptors for object recognition using the same pipeline described above. In the context of object manipulation by robots, (Costanzo. et al., 2018) proposed a hybrid and real-time object recognition system.

The second category of pipeline consists in using deep learning techniques which offer better recognition rates but requires high performance computational hardware. (Zaki et al., 2016) and (Schwarz et al., 2015) have introduced systems based on CNN (Convolutional Neural Network) architectures. (Guo et al., 2020) also presented an interesting comparison of deep learning based systems in the context of recognition from point cloud data. Among them are the following architectures: PointNet++, (Qi et al., 2017), VoxNet (Maturana and Scherer, 2015), etc.

To our knowledge, most of the existing methods which allow processing point cloud data in the context of object recognition or vehicle localization, have as main focus the improvement of the accuracy. In our case, we are more interested in enabling and improving the real-time capability of the proposed system.

3 PROPOSED SYSTEM

In this paper, we introduce a system allowing automatic recognition of common everyday objects from point cloud data. After validation, it is intended to be used for assisting bus drivers by performing accurate localization even under adverse climatic conditions. As shown in Figure 2, the system includes several processing blocks which we describe in this section.

3.1 Preprocessing

The number of points per point cloud varies and since we are working with local 3D descriptors, the size of the generated representation varies as well. Moreover, the some next processing blocks, namely dimension reduction and classification require a constant feature vector size. In order to deal with this issue, we propose two different operating modes called normal and cluster.

In the normal mode, the system applies downsampling which aims to reduce the size of the point cloud data. Basically, the employed technique creates a 3D voxel grid. Then, in each voxel, all the points present will be approximated with their centroid. In the context of the object dataset that we are using for evaluation, we set empirically the parameter that sets the
size of each voxel to $d = 0.5 \text{ cm}$. This step is important since it reduces the computing time by getting rid of redundant data. Even with such an approach, the size of the input vectors remains variable. To compensate for this, we apply a simple zero padding operation after performing the feature extraction step.

As shown in Figure 3, the cluster mode applies the $k$-means algorithm to the point cloud data (in blue). It consists in an unsupervised machine learning technique that identifies similar instances or points and assigns them to a cluster (in red) using a specific metric (Euclidean distance). After applying $k$-means, the closest point (black circles) to the centroid (green squares) is selected for each cluster. In this case, downsampling is not applied for the sake of keeping all the details. Thus, for each one of the selected keypoints, a local descriptor will be computed in the next step based on all the neighboring points within the defined radius. Similarly as the normal mode, the number of clusters $k$ involved in the $k$-means algorithm needs to be set empirically ($k = 50$ for the object dataset used for evaluation).

### 3.2 Feature Extraction

In order to optimize the performance of the system in terms of recognition, an efficient representation of the input is required. The proposed system allows the extraction of two distinct local 3D descriptors namely USC (Tombari et al., 2010) and SHOT (Salti et al., 2014) (including the color version SHOT-rgb). Since the generation of both representations implies that the neighbors around the concerned point be taken into account, we need to set empirically the search radius parameter (in our case $r = 1.0 \text{ cm}$).

Both descriptors generate an histogram for each point and the size of these descriptors, related to the histogram number of bins, is different. More precisely, here is the size of the related histograms: $|V_{USC}| = 1980$, $|V_{SHOT}| = 352$ and $|V_{SHOT\_rgb}| = 1344$ bins. The final feature vector consists of a concatenation of all histograms. However, this last operation varies depending on the selected mode. In the case of normal mode, the concatenation is performed on all the histograms generated from all the points after downsampling. Then, since the number of points is not the same, a zero padding operation is applied. For the cluster mode, the number of keypoints is always the same as initially defined. Therefore, the concatenation is performed only on the histograms generated from these keypoints.

### 3.3 Dimension Reduction

The size of the generated representation using the chosen 3D descriptors namely USC and SHOT (in-
including the color version SHOT-rrgb) is huge considering the number of keypoints and the size of histograms. Therefore, we need to reduce its size to ease and optimize the recognition performance. Even if several dimension reduction techniques exist, the most commonly used is the PCA (Principal Components Analysis) (Jolliffe and Cadima, 2016). It is defined as an unsupervised, non-parametrical statistical technique used in machine learning aiming to identify the hyperplane such that the variance of the data when projected onto this plane is changed as little as possible. The generated axes or principal components are orthogonal. The reduced feature vector which consists in a certain number of principal components feeds the classification block.

Taking into account the number of samples and the size of previously generated representations, applying the standard version of the PCA might lead to memory issues. Therefore, we propose to use the incremental or online version which allows to apply PCA in an iterative way by taking as input small batches of samples (from the dataset).

### 3.4 Classification

In order to perform recognition based on the reduced representation, we use a classification technique. It consists in a supervised machine learning technique which requires a learning phase using labeled data. Even if several techniques might be used, we adopt the multi-class and linear SVM (Support Vector Machine) since it yields relatively good performance. It has been proposed by (Cortes and Vapnik, 1995) as a binary classifier aiming to find a hyperplane to optimally separate two distinct classes. It might be defined by \( y_i = \text{sign}(w \cdot x_i + b) \) where the maximum-margin hyperplane is represented by \((w, b)\), the feature vectors by \(x_i \in \mathbb{R}^n\) and labels by \(y_i \in \{\pm1\}\). In order to distinguish between several objects, we adopt the One-Against-All architecture which consists of several linear binary-SVM classifiers.

Similarly to the PCA and in order to deal with huge amount of samples, we adopt the online version of the multi-class SVM.

### 4 EVALUATION

In order to evaluate the proposed system in both modes namely normal and cluster, we used the benchmark and publicly available RGB-D\(^1\) dataset (Lai et al., 2011) (see Figure 4). It consists of 300 common everyday objects organized in 51 distinct categories. The samples are collected using a Kinect-style camera enabling the emphasis of depth information. The type of sensors employed to collect the used dataset is not the same as the one we are targeting and which consists in LiDAR sensors. However, the data format is the same since the LiDAR sensors also collect point cloud data. The main differences when using these two types of sensors lies the covered range and density. Moreover, the main aim of this work consists in validating the systems and defining the most suitable configuration in terms of mode, descriptor and algorithm version. Therefore, in order to speed up the evaluation, we selected three different stratified subsets namely: first subset (greens, pliers, food jar, hand towel and toothpaste), second subset (binder, sponge, camera, kleenex and lemon) and third subset (water bottle, scissors, banana, flashlight and bowl). Each one contains 250 different samples. The choice of the different objects and samples for each subset is made in a random manner.

As for the evaluation metrics, we employed several ones. Usually, the most important one is related to the recognition rate. Thus, we compute the accuracy which represents the number of correct predictions divided by the total number of samples. Moreover, since the proposed system performs multi-class classification, the recognition rate for each object is also provided. The accuracy might be biased if the system performs better with certain classes than for the others. Therefore, by computing the average of all the objects recognition rate, we provide more precision about the system performances. Furthermore, since we are targeting real-time application, we also measured the elapsed time when performing learning and evaluation with different system configurations.

As for the evaluation protocol, we adopted the k-fold cross-validation splitting strategy. We set the number of folds to \(k = 5\) and therefore, each one of the three sets of data (namely first subset, second subset and third subset) is divided into five subsets. During each iteration, four subsets are used for training and
the last one for evaluation. The final accuracy of each set of data is computed by averaging the obtained accuracy at each iteration.

5 RESULTS AND DISCUSSION

In this section, we present and discuss the results obtained after performing several evaluations. Each one focusing on a specific metric.

Figure 5 displays system performance in terms of accuracy when using the two local 3D descriptors in both working modes (normal and cluster). We notice that the highest accuracy is achieved by the SHOT-rgb descriptor in both modes. Indeed, it allows to reach 87.55% and 71.54% in normal and cluster mode, respectively. The reason might be explained by the additional color information provided by SHOT-rgb. As for the USC descriptor, it yields the lowest accuracy.

In Figures 6a and 6b is shown the variation of accuracy for the two local 3D descriptors in both working modes (normal and cluster) regarding the number of principal components. Similarly to Figure 5, the best performance is achieved by the SHOT-rgb descriptor in normal mode. We also notice how the PCA contributes to compress the feature vectors and highlights the relevant and discriminant information. Thus, we are able to obtain a relatively good accuracy using only a few attributes or principal components. It might be explained by the fact that adding more attributes or variables does not necessarily improve the accuracy since some of them introduces noise to the classifier.

Table 1 provides details on the recognition rates for each subset and objects. We notice that the SHOT-rgb descriptor achieves relatively high accuracy with 92.17% in the second subset. As for the recognition rate for each object, we notice that for certain objects (pliers, lemon, hand_towel), the highest accuracy of 100.00% is achieved. Based on the obtained results in the different subsets, there is an average difference of $\approx 8.00\%$ between the normal and cluster mode.

In Table 2 are shown the results relevant to the computational efficiency of the proposed system. One
Table 1: Accuracy comparison between the three subsets & both modes.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Descriptors</th>
<th>Objects recognition rates (%)</th>
<th>Average (%)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>greens</td>
<td>pliers</td>
<td>food_jar</td>
</tr>
<tr>
<td>First subset</td>
<td>Normal</td>
<td>SHOT</td>
<td>95.83</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td>SHOT</td>
<td>47.92</td>
<td>98.15</td>
</tr>
<tr>
<td>Second subset</td>
<td>Normal</td>
<td>SHOT</td>
<td>95.83</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td>SHOT</td>
<td>85.42</td>
<td>98.15</td>
</tr>
<tr>
<td>Normal</td>
<td>USC</td>
<td>81.25</td>
<td>100.00</td>
<td>55.56</td>
</tr>
<tr>
<td>Cluster</td>
<td>USC</td>
<td>31.25</td>
<td>92.39</td>
<td>81.48</td>
</tr>
<tr>
<td>Third subset</td>
<td>Normal</td>
<td>SHOT</td>
<td>74.07</td>
<td>75.00</td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td>SHOT</td>
<td>42.59</td>
<td>27.78</td>
</tr>
<tr>
<td>Normal</td>
<td>SHOT</td>
<td>74.07</td>
<td>94.44</td>
<td>94.44</td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td>SHOT</td>
<td>72.22</td>
<td>33.33</td>
</tr>
<tr>
<td>Normal</td>
<td>USC</td>
<td>74.07</td>
<td>55.56</td>
<td>31.48</td>
</tr>
<tr>
<td>Cluster</td>
<td>USC</td>
<td>59.26</td>
<td>0.00</td>
<td>64.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>water_bottle</td>
<td>scissors</td>
<td>banana</td>
</tr>
<tr>
<td>Normal</td>
<td>SHOT</td>
<td>77.78</td>
<td>81.25</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td>SHOT</td>
<td>50.00</td>
<td>81.25</td>
</tr>
<tr>
<td>Normal</td>
<td>SHOT</td>
<td>83.33</td>
<td>83.33</td>
<td>68.75</td>
</tr>
<tr>
<td></td>
<td>Cluster</td>
<td>SHOT</td>
<td>77.78</td>
<td>87.50</td>
</tr>
<tr>
<td>Normal</td>
<td>USC</td>
<td>38.89</td>
<td>52.08</td>
<td>41.67</td>
</tr>
<tr>
<td>Cluster</td>
<td>USC</td>
<td>57.41</td>
<td>91.67</td>
<td>68.75</td>
</tr>
</tbody>
</table>

of the main differences between the normal and cluster modes lies in the feature vectors size. Indeed, the latter mode is less complex since the descriptor vector size is relatively smaller than the former one. The smallest vector size is associated with SHOT. Indeed, it is 3.8× and 5.5× smaller than SHOT-rgb and USC, respectively. The other criteria that is considered is the computing time. The SHOT descriptor takes the less time in comparison to the others. As for the online version (see the second line for each descriptor in Table 2), even if it deals with the memory issue, it slightly reduces the accuracy and increases the computing time.

Table 2: Computing time and vector size comparison.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Descriptors</th>
<th>Time (ms)</th>
<th>Vector Size</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>SHOT</td>
<td>38</td>
<td>574112</td>
<td>82.55</td>
</tr>
<tr>
<td></td>
<td>SHOT_rgb</td>
<td>65</td>
<td>2192064</td>
<td>87.00</td>
</tr>
<tr>
<td></td>
<td>USC</td>
<td>100</td>
<td>3196760</td>
<td>73.98</td>
</tr>
<tr>
<td>Cluster</td>
<td>SHOT</td>
<td>206</td>
<td>17600</td>
<td>59.33</td>
</tr>
<tr>
<td></td>
<td>SHOT_rgb</td>
<td>241</td>
<td>67200</td>
<td>43.12</td>
</tr>
<tr>
<td></td>
<td>USC</td>
<td>657</td>
<td>98000</td>
<td>54.06</td>
</tr>
</tbody>
</table>

Taking into account all the evaluation criteria, we suggest that the most suitable system needs to combine the SHOT descriptor and the online versions of both PCA and multi-class SVM.

6 CONCLUDING REMARKS

In this paper, we introduce an object recognition system from point cloud data. We compare two local 3D descriptors namely USC and SHOT (including the color version SHOT-rgb) and show through experiments that the latter performs better. We also propose two different modes namely normal and cluster. The former one performs better in terms of accuracy, but the second one allows to reduce considerably the size of the input point cloud. We overcome the memory issue when working with huge amount of data by using incremental versions of PCA and multi-class SVM. As further works, we are planning on using the proposed system on LiDAR data in the context of vehicle localization. We also intend to optimize the proposed system in order to increase its performances.

ACKNOWLEDGEMENTS

This work was supported by NSERC Collaborative Research and Development grant CRDPJ 511843 – 17. We also acknowledge the providers of the used benchmark RGB-D Object Dataset (Lai et al., 2011).
REFERENCES


