Semi-supervised Online Learning for Efficient Classification of Objects in 3D Data Streams

Ye Tao Rudolph Triebel Daniel Cremers

***Abstract*— We present a novel learning algorithm especially designed for challenging, large-scale classification problems in mobile robotics. Our method addresses two important aims: first it reduces the required amount of interaction with a human supervisor, which increases the level of autonomy of the learning process. And second, it has the capability to update its internal representation online with every new observed data sample, which makes it adaptive to new environments. The proposed method is based on a combination of two established methods, namely Online Star Clustering and Label Propagation, but it extends and modifies these in such a way that significant shortcomings such as classification inaccuracy and run time inefficiency can be resolved. In experiments on large benchmark data sets, we show that our approach can quickly learn to classify 3D objects with a significantly reduced amount of required ground truth labels for training.**

I. Introduction

With the advent of fast, high-resolution and at the same time affordable 3D sensors, mobile robotic platforms have recently experienced an enormous progress in terms of their perceptual capabilities. RGB-D cameras have become a standard equipment for mobile robots, and the recent achievements in RGB-D sensing suggest that these sensors will be the main exteroceptive sensor source in the near future. Their impact mainly stems from their high resolution, evidenced in very densely sampled resulting 3D point clouds, and from their ability to sense depth and color simultaneously and at high frame rates. However, while depth sensors are also used widely in other application areas such as surveillance tasks, in mobile robots there are at least two specific major challenges to solve. First, the large amounts of data produced by these sensors places particular problems for the learning algorithms used for automated semantic annotation tasks such as 3D object classification or semantic mapping. In principle, the major design goal for a mobile robotic platform is to be as autonomous as possible, i.e. interactions with human supervisors should be reduced to a minimal amount[1], and this includes interactions needed for learning, e.g. when labeling ground truth data for training. Therefore, to be autonomous, a robot should ask for semantic information only rarely, but this is hard when the acquired amount of data is large. And second, mobile robots usually operate in frequently changing environments, and they need to take decisions quickly and based on their current situation. Thus, the employed learning algorithms need to be *adaptive*, which means that many standard offline learning techniques are inadequate due to their computational requirements and their inability to modify internal representations on the fly.

Therefore, in this paper we propose a learning algorithm that simultaneously reduces the required amount of human effort in terms of providing ground truth label information, and operates online in the sense that it incorporates new information directly to update and refine its internal representation. The result is a fast and effective learning method that is particularly suited for semantic annotation of large 3D data streams, as we will show in the experiments. We achieve that using a novel online clustering algorithm that is particularly taylored for semi-supervised learning. Inspired by established graph-based methods such as Online Star Clustering [1] and online Affinity Propagation [2], our approach also uses an undirected, bipartite graph, however with the difference that non-exemplar nodes can not turn into exemplar nodes during vertex insertion, and that we use a more efficient nearest-neighbor search when inserting a new vertex. This and some other modifications make our clustering method more useful for subsequent label propagation, a fast and effective semi-supervised learning method. As we show in experiments on standard benchmark data, our method is able to efficiently learn 3D objects from large data streams online and with only little input from a human supervisor.

II. Related Work

Our work combines semi-supervised learning (SSL) with online clustering and is therefore mostly related to these two areas[2]. For SSL, there is a good overview text book edited by Chapelle *et al.* [3], where detailed theoretical background is given, as well as a description of the most common techniques, including transductive Support Vector Machines (tSVM) [4], Gaussian Process classification (GPC) with null-category noise model [5] and Label Propagation [6]. Our proposed method mostly relates to the latter one, mainly due to efficiency reasons and because, as a graph-based approach, Label Propagation is very well suited for combination with efficient graph-based online clustering methods, which we employ in our approach. In that context, a number of earlier works have been proposed, where the most relevant ones are online k-means clustering [7], [8], online Expectation Maximization (EM) [9], [10], online affinity propagation (AP) [2], and online star clustering (OSC) [1]. The latter two approaches have the big advantage over k-means and EM

All authors are with Dep. of Computer Science, Technical University of Munich, Boltzmannstrasse 3 85748 Garching, Germany [ye.tao, rudolph.triebel, daniel.cremers]@in.tum.de

[1]Note that this is different from interactions with human *users*, where the robot itself provides a service and does not depend on human input.

[2]Note however the difference to *self-supervised* learning where semantic information comes from a different sensor source (see, e.g. [17])

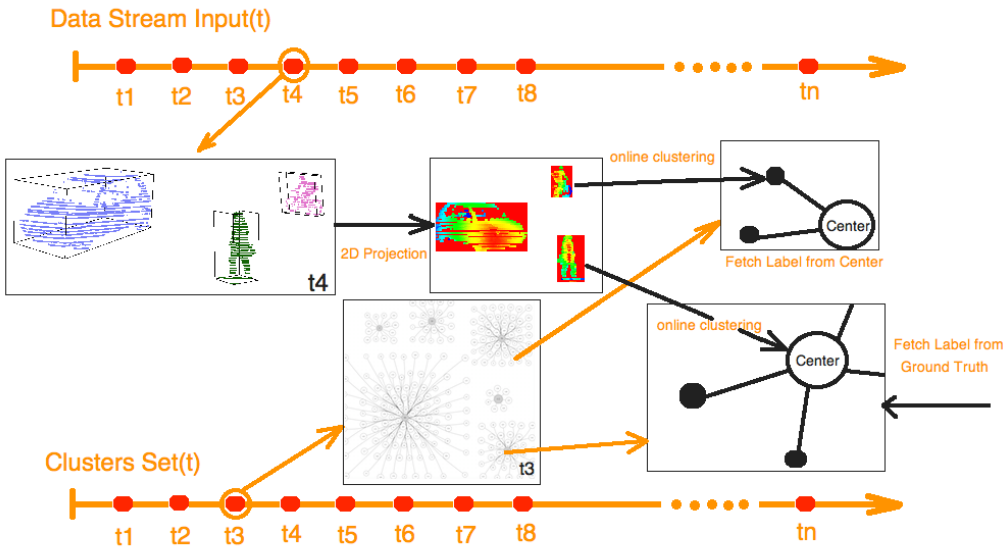


Fig. 1: Overview of our online learning approach for 3D object classification from data streams. See text for details.

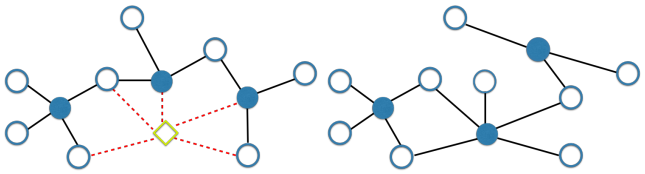
that they do not require the number of clusters to be specified beforehand. Instead, they use a similarity threshold and determine the number of clusters implicitly. Our approach builds on OSC because it is more efficient than AP clustering (see also [11] for an application of OSC to unsupervised scene classification). However, as we will show, OSC has some drawbacks for our application in semi-supervised learning.

To compute features from 3D input data, we use Hierarchical Matching Pursuit (HMP) [12], an unsupervised learning algorithm based on sparse coding. In the area of unsupervised feature learning, many different approaches have been proposed, including Restricted Boltzmann Machines (RBM) [13], convolutional deep belief networks [14] and denoising autoencoders [15]. However, recent results on HMP-based classification [16], [12] show that they are very powerful and at the same time comparably efficient in learning. Therefore, we decided to use HMP features for our work.

III. Overview

Fig. 1 shows a schematic overview of our online 3D object classification method. We see two time lines: one on the top, which represents the incoming 3D point clouds at each time step, and one at the bottom for the status of the internal graph representation per time step. The depicted situation consists of an existing cluster graph at time step t_3 and a new sensor observation (point cloud) from the next time step t_4. In our pipeline, we first compute Hierarchical Matching Pursuite (HMP) features [16] for each pre-segmented point cloud in the current frame. We assume that the individual 3D objects are segmented with their 3D bounding box. Such a segmentation can be obtained from a tracking algorithm that separates moving objects from the static scene parts or by ground plane segmentation (see for example [18]). We note that, depending on the 3D sensor a pre-processing step might be required before feature computation. In the example here, data is obtained from a 3D laser scanner, which means that we have to create depth images from the point clouds before being able to apply HMP feature computation. Examples are shown in the center box in the upper part of the figure. Of course, when using RGB-D cameras, depth images are already available and need not to be computed explicitly.

The obtained HMP feature vectors are then inserted as vertices into the cluster graph. This graph distinguishes between center and satellite vertices, where the centers are *exemplars* for the satellites connected to them (details follow in Sec. V). Thus, a newly added vertex can either end up as a satellite of an already existing center, or it can be itself a new center. This is exemplified in the figure with the pedestrian and the cyclist, where the former builds a new center vertex, and the latter is associated to a new satellite. Then, the algorithm queries ground truth labels for the new centers if there are any, and infers class labels for the remaining vertices using Label Propagation. Our reasoning behind this is that centers are good potential representatives of an object class, particularly if they have many satellites attached, which by construction of the graph are similar to them. Thus, propagating labels from centers to satellites will lead to less misclassifications and fewer label queries than, e.g. propagating from satellites to centers. Note that the number of centers directly influences the performance of the algorithm: fewer centers lead to less label queries, making the learning algorithm more autonomous, but at a higher chance of misclassifications as more satellites will be different from their centers, i.e. the clusters will be less *pure*. The challenge is therefore to obtain pure, but few clusters (centers) at the same time. In Sec V we show how we address this trade-off, but first we consider a different approach combining two standard methods, and we show the drawbacks there that motivate our own method.



(a) vertex insertion (b) updated star graph

Fig. 2: Vertex insertion in the standard Online Star Clustering algorithm [1]. **(a)** A new vertex is inserted (yellow diamond) and all neighbors that are more similar than ϑ are determined (dashed lines). In this case, this results in a new center, because the degree of the new node is higher than those of the adjacent centers. **(b)** Rearranging requires removing and adding some edges and changing the role of some vertices.

IV. Online Semi-Supervised Learning

As a starting point, and also as a baseline for comparison with our proposed method, we consider here a straightforward combination of two concrete algorithms: Online clustering using the algorithm of Aslam *et al.* [1] and subsequent semi-supervised learning using Label Propagation [19].

A. Online Star Clustering

The main idea of the Online Star Clustering (OSC) algorithm [1] is to find a minimal number of maximal star-shaped subgraphs from a given thresholded similarity graph ("min-max criterion"). This means that the algorithm starts with a graph G_ϑ that consists of nodes v_i for each data sample and edges e_{ij} connecting two nodes that are more similar than a given threshold ϑ, i.e. $s(v_i, v_j) \geq \vartheta$ where s is a similarity measure. Then, it identifies some vertices as cluster *centers* and the remaining ones as *satellites* and removes all edges from G_ϑ that connect two satellites or two centers. The assignment of centers and satellites is made such that the number of clusters is minimal and the cluster sizes are maximal, and cluster centers have a higher degree (number of incident edges) than their connected satellites. For our application, the OSC algorithm has two major advantages over other clustering methods: First, it does not require the number of clusters to be given. Instead, its only parameter is the similarity threshold ϑ, which implicitly influences the number of resulting clusters. And second, the creation of the graph can be done online, i.e. after insertion of a single new vertex the min-max criterion is still valid. To guarantee this, in some cases the algorithm needs to re-assign centers and satellites and also to remove and add edges. An example of this is shown in Fig. 2.

The original OSC algorithm uses the cosine distance as a similarity measure between two connected vertices v_i and v_j, i.e.

$$s(v_i, v_j) = \frac{\langle v_i, v_j \rangle}{\|v_i\| \|v_j\|} = \cos(\alpha), \tag{1}$$

where α is the angle between the vectors that correspond to v_i and v_j. The authors show that with this similarity measure the similarity between two satellite vertices $u_i^{j_1}$ and $u_i^{j_2}$ that are connected to the same center c_i is bounded by

$$s(u_i^{j_1}, u_i^{j_2}) = \cos\gamma \geq \cos\alpha_1 \cos\alpha_2 + \frac{\vartheta}{1+\vartheta} \sin\alpha_1 \sin\alpha_2, \tag{2}$$

where γ is the angle between the two satellites and α_1 and α_2 are the angles between the satellites and the center c_i. The interesting thing about this formulation is that it can be completely expressed in terms of dot products between feature vectors, provided that these are normalized. That means, we can also use other similarity measures instead by replacing dot products with *Mercer kernels*, for example the Gaussian kernel

$$k(v_i, v_j) = \exp\left(-\frac{\|v_i - v_j\|^2}{2\sigma^2}\right) \tag{3}$$

with a variance parameter σ, or the inverse city block kernel

$$cb(v_i, v_j) = \frac{1}{\sum_k |v_{ik} - v_{jk}| + \xi} \tag{4}$$

whith a constant parameter ξ. Thus, we then obtain a *kernelized* OSC algorithm. In the experiments (Sec. VI), we show that an appropriate kernel can substantially improve the measure of similarity.

B. Label Propagation

The OSC algorithm is a classical unsupervised learning method, i.e. it does not incorporate ground truth information for learning. However, in our application, we aim for automated semantic annotation, and this information can only come from some human supervisor. Therefore, we combine clustering with a semi-supervised learning (SSL) method by assigning ground truth labels to the center vertices and infering the labels for the unlabeled satellite vertices. In particular, we use Label Propagation (see [19], Algorithm 11.2). This method first computes an affinity matrix W where the entries are the node similarities, i.e. $w_{ij} = s(v_i, v_j)$ and it sets $w_{ii} = 0$. It then chooses a parameter $\alpha \in (0, 1)$ and a small $\epsilon > 0$ and computes $\mu := \alpha/(1-\alpha)$. With this, it iterates over all vertices and updates the labels $\mathbf{y}_i$ of the vertices v_i in every iteration. We model class labels as vectors of a fixed length K, which determines the number of classes. A vertex v_i has then the class label k if $y_i^k = 1$ and all other entries of the vector $\mathbf{y}_i$ are zero. For unlabeled vertices, $\mathbf{y}_i$ is equal to the zero vector. The operations performed in each iteration of Label Propagation are as follows: If v_i is a labeled (center) node with an associated ground truth label $\mathbf{y}_i^{(0)}$ then the update rule at iteration $t = 1, 2, \ldots$ is

$$\mathbf{y}_i^{(t+1)} \leftarrow \frac{\sum_j W_{ij} \mathbf{y}_j^{(t)} + \frac{1}{\mu} \mathbf{y}_i^{(0)}}{\sum_j W_{ij} + \frac{1}{\mu} + \epsilon}. \tag{5}$$

If v_i is an unlabeled satellite vertex, then the rule is

$$\mathbf{y}_i^{(t+1)} \leftarrow \frac{\sum_j W_{ij} \mathbf{y}_j^{(t)}}{\sum_j W_{ij} + \epsilon}. \tag{6}$$

These rules are computed until a convergence criterion is reached. The described Label Propagation (LP) algorithm

is formulated as an offline algorithm, although one could think of an extension to the online case. However, due to the shortcomings of this combined 'OSC+LP' approach, which we describe next, we do not consider an online LP version, but instead suggest an improved algorithm in Sec. V.

C. Drawbacks of the OSC+LP Approach

The presented combination of OSC and LP has at least three major drawbacks: First, it requires a nearest-neighbor search for each newly inserted vertex over the entire existing data set. This increases the run time significantly when the data set is very large. Second, due to the requirement that the min-max-criterion has to be fulfilled always, it can lead to many changes from satellite vertices to center vertices and vice-versa. This is not only inefficient in terms of computation time, but it also causes more label queries, especially if satellites turn into centers. In a sense, the fact that OSC guarantees a minimal number of clusters remedies this somehow, because with fewer clusters there are less label queries. However, as we perform label queries in every time step, one has to consider all vertices that at some point in the past have been centers, i.e. this includes all those satellites that where flipped from centers. And the third problem with OSC+LP is that the algorithm uses a fixed similarity threshold ϑ, which results in many isolated vertices that are not connected to any other one. These outliers again increase the required number of label queries, because each of them is considered a center of a cluster with size 1. In the next section, we propose our improved version of the algorithm, which particularly mitigates these three drawbacks.

V. Proposed Approach

Assume we are given a stream of data points $\mathbf{x}_1, \mathbf{x}_2, \ldots$ with $\mathbf{x}_i \in \mathbb{R}^d$. Our aim is to incrementally build from these data a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{Y}, P, S)$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ are the edges, $\mathcal{Y}$ are the class labels for each vertex, P are the *properties* of the vertices and S are the *similarities* assigned to each edge. Concretely, each vertex $v_i \in \mathcal{V}$ can have either the property 'center' or 'satellite', and each edge $e_{ij} \in \mathcal{E}$ connecting vertices v_i and v_j has a similarity value s_{ij} based on a distance measure between v_i and v_j. The center vertices play the role of *exemplars*, i.e. they are representatives of a whole set of other vertices, namely the satellites connected to them. In any stage of the algorithm, the graph G is bipartite, i.e. there are only edges connecting centers with satellites, and never edges between two centers nor between two satellites. The main idea of our semi-supervised online learning algorithm is to use only the centers to query ground truth class labels and to use these to infer the labels for the satellites, i.e. in a similar way as is done in label propagation. The individual steps of our algorithm are described next.

A. Vertex Insertion

As mentioned, our algorithm is inspired by the Online Star Clustering (OSC) approach [1]. This means, we also aim for a vertex insertion method that is efficient and at the same time produces graphs that correspond to a good data clustering. However, in standard OSC, the major focus is laid more on the latter by guarateeing that after insertion the graph still consists of a minimal set of maximal star-shaped sub-graphs (i.e. clusters). While this is a good property for pure unsupervised learning applications, in semi-supervised learning as we propose it, it is more advantageous to have *purer* clusters, even if the number of clusters is not minimal. Concretely, this means that we do not stricly connect a new vertex to its closest neighbors in the graph as in OSC, thereby accepting that the number of clusters can be sub-optimal. This slight drawback is however outweighed by the fact that our insertion is more efficient and produces purer clusters.

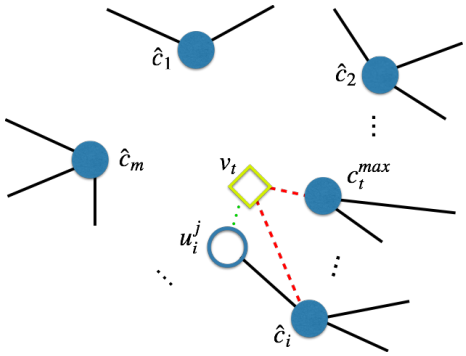


Fig. 3: Vertex insertion into the graph G. At time t, we insert vertex v_t (yellow diamond). For that, we first compute the m center vertices $\hat{c}_1, \ldots, \hat{c}_m$ that are closest to v_t (blue filled circles). Each of these centers has satellite vertices attached, here indicated with lines. The center closest (or most similar) to v_t is denoted c_t^{max}. Then, for $i = 1, \ldots, m$ we compute the similarities $s(v_t, \hat{c}_i)$ and $s(v_t, c_t^{max})$ (dashed red lines), as well as the similarities between v_t and all satellites u_i^j attached to center $\hat{c}_i$ (dotted green line). All such satellites that fulfill the condition in Eq. (8) are detached from $\hat{c}_i$ and connected to v_t, which then becomes a center vertex.

For the description of our insertion method, we define the set of centers $\mathcal{C}_t$ at time t as $\mathcal{C}_t = \{v_i \in \mathcal{V}_t \mid p_i = \text{'}center\text{'}\}$ and its cardinality as C_t. With this, the first step of insertion is to find a fraction $q \in [0, 1]$ of centers in $\mathcal{C}_{t-1}$ that are most similar to the new vertex v_t. This has two advantages: First, it does not require a fixed similarity threshold for connecting vertices, which avoids un-connected outliers. And second, it is more efficient than finding neighbors in the entire set of vertices $\mathcal{V}_t$, because $\mathcal{C}_t$ is usually much smaller than $\mathcal{V}_t$. Thus, formally we find an index ordering $\pi : \mathbb{N} \to \mathbb{N}$ so that $s(c_{\pi(i)}, v_t) > s(c_{\pi(i+1)}, v_t)$ for all $i = 1, \ldots, C_t - 1$, where we denote the similarity s as a binary function of vertices, and the elements of $\mathcal{C}_t$ are $c_1, \ldots, c_{C_t}$. Furthermore, we define the center that is most similar to v_t as c_t^{max}, i.e. $c_t^{max} := c_{\pi(1)}$. Then, the result of this first step is a subset $\hat{\mathcal{C}}_t \subset \mathcal{C}_t$ consisting of the first m sorted centers, i.e. $c_{\pi(1)}, \ldots, c_{\pi(m)}$ where $m = \lfloor qC_t \rfloor$.

In the next step, we search for neighbors of v_t in the set of satellites that are connected to centers in $\hat{\mathcal{C}}_t$. Thus, for

each of the m most similar centers $\hat{c}_1, \ldots, \hat{c}_m$ we loop over all attached satellites u_i^j and decide whether they should be detached from their center $\hat{c}_i$ and reconnected to v_t, which then becomes a new center, or whether v_t is simply added as a satellite to c_t^{max}. Our criterion for this re-connection step is based on the similarity of the satellites u_i^j and v_t, but also on the general similarity of the cluster represented by $\hat{c}_j$. Concretely, we compute the *normalized similarity*

$$\bar{s}(v_t, \hat{c}_i) := \frac{s(v_t, \hat{c}_i)}{s(v_t, c_t^{max})} \tag{7}$$

and do a re-connection whenever

$$s(v_t, u_i^j)\bar{s}(v_t, \hat{c}_i) > s(u_i^j, \hat{c}_i). \tag{8}$$

If the condition in Eq.(8) is not valid for any satellite u_i^j, we connect v_t as a new satellite to c_t^{max}.

B. Minimal Average Cluster Size

As mentioned in Sec. IV-C, one major problem with standard OSC is that it tends to produce many unconnected vertices, which are treated as centers. This means that the SSL algorithm will query labels for them, even though the obtained information is often not very useful, because the queried labels are not representative for many other samples. The same problem occurs in our approach when many satellites of a given cluster center are detached and reconnected as described. Therefore, we introduce a parameter ω_{min}, which defines a lower bound on the average cluster size. Then, each time a satellite u_i^j should be detached from its center $\hat{c}_i$ according to Eq. (8), we test whether the average cluster size $\bar{z}_t$ is still larger than w_{min}. If this is not the case, all vertices of the cluster represented by $\hat{c}_i$ are inserted into the cluster that is closest to $\hat{c}_i$. Note that $\bar{z}_t$ at time t can be easily computed from $\bar{z} = \frac{t}{C_t}$, because at time t there are t vertices inserted in total and the number of clusters is C_t. The same procedure is done with the potentially new cluster formed by v_t as a center: As long as adding this new cluster does not cause the average cluster size $\bar{z}$ to be smaller than ω_{min}, it can be added. Otherwise, it is not added and v_t is attached to its closest center, as above. To avoid unnecessary re-connecting steps, we therefore wait until all centers closest to v_t are processed before we actually perform the reconnection of satellites.

C. Label Propagation

As in the OSC+LP approach described above, our last step is also label propagation. That is, if the newly added vertex v_t ends up as a new center, we query a ground truth label for it and propagate this new label to the satellites attached to v_t. If v_t has become a satellite, we propagate the label of the corresponding center (which was queried earlier queried) to it. Here, we note an important difference to the OSC algorithm: in our approach, all satellite vertices are connected to exactly one center. Therefore, Eq. (6) simplifies to a simple "copy" of the label from the center to the satellites. Similarly, Eq. (5) directly assigns the ground truth label to the labeled center. Thus, by our graph construction, the LP method can also be performed more efficiently.

D. Time-dependant Paramters

Two main parameters of our algorithm are the fraction q of most similar centers considered and the lower bound ω_{min} on the average cluster size. For both, there is an implicit dependence on the current size of the graph. In the beginning, there are only few samples and the graph is sparse. Thus, the fraction q of nearest neighbors can be larger, because nearest-neighbor search will anyhow be very efficient. Similarly, the minimal number of elements ω_{min} of an average cluster can be larger when there are more vertices in the graph. Therefore, we recompute q and ω_{min} at time step t as

$$\omega_{min}^t = \omega_{min}(1 - e^{-\tau * t}) \tag{9}$$

$$q^t = q(1 - e^{-\tau * t}), \tag{10}$$

where τ is a damping parameter.

Algorithm 1 summarizes all steps of our approach[3]. Note that satellites are not actually disconnected until all centers in $\hat{\mathcal{C}}$ have been considered and the new vertex forms a cluster that is large enough (line 15-17). Clusters that are too small are removed and all elements are assigned to the cluster that is most similar to their center. This is the $Recluster$ step.

Algorithm 1: Online Exemplar Based Clustering

Data: stream $\mathbf{x}_t$ for $t = 1, 2, \ldots$
Input: nearest-neighbor fraction q, damping factor τ, min average cluster size ω_{min}
Output: inferred or queried labels y_t

1 $v^t \leftarrow CreateVertex(\mathbf{x}^t)$
2 $\omega_{min}^t \leftarrow \omega_{min}(1 - e^{-\tau t})$ (see Eq.(9))
3 $q^t \leftarrow q(1 - e^{-\tau t})$ (see Eq.(10))
4 $\hat{\mathcal{C}}_t \leftarrow MostSimilarExemplars(\mathcal{C}^t, v^t, q)$
5 $\bar{z} \leftarrow \frac{t}{C_t}$
6 $\mathcal{K} \leftarrow \emptyset$
7 **forall the** $\hat{c}_i \in \hat{\mathcal{C}}_t$ **do**
8 **forall the** u_i^j *connected to* $\hat{c}_i$ **do**
9 **if** *Eq.* (8) *is true* **then**
10 $\mathcal{K} \leftarrow \mathcal{K} \cup u_i^j$
11 $MarkDisconnected(u_i^j, c_i)$
12 **if** $\bar{z}_t > \omega_{min}^t$ & $Size(\hat{c}_i) < \omega_{min}^t$ **then**
13 $Recluster(\hat{c}_i, \{u_i^j\})$
14 **if** $\bar{z}_t > \omega_{min}^t$ & $|\mathcal{K}| + 1 > \omega_{min}^t$ **then**
15 $DisconnectAllMarked(\{u_i^j\}, \{\hat{c}_i\})$
16 $MakeNewCluster(v_t, \mathcal{K})$
17 $y_t \leftarrow QueryGroundTruthLabel(v_t)$
18 $y_t \leftarrow PropagateLabels(v_t, \mathcal{K})$
19 **else**
20 $UnmarkAll(\{u_i^j\}, \{\hat{c}_i\})$
21 $Connect(v_t, c_t^{max})$
22 $y_t \leftarrow PropagateLabel(c_t^{max}, v_t)$

[3]An implementation of our approach in C++ is available at `https://github.com/mmmonkeytao/oscl.git`.

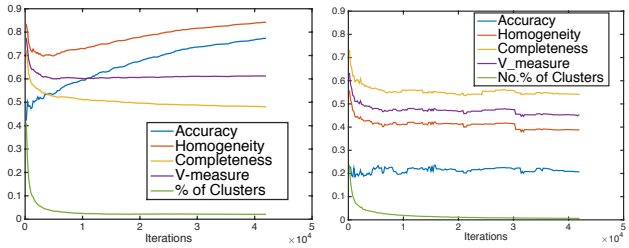


Fig. 5: Evaluation on the RGB-D data set based on V-measure (violet), classification accuracy (blue) and number of clusters per vertex (green). **Left:** Our approach. **Right:** OSC+LP. Note that our approach significantly outperforms the OSC+LP method in terms of accuracy and V-measure, although there is no big difference in the number of clusters.

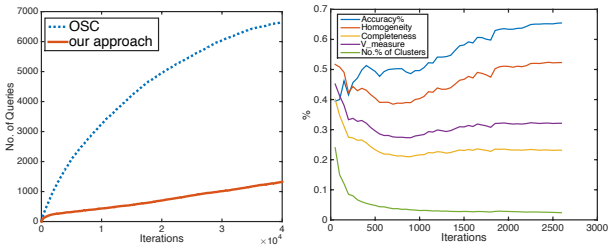


Fig. 6: **Left:** Accumulated number of generated label queries. Our approach generates significantly less label queries. **Right:** Result of our algorithm on the KITTI data set. The accuracy is worse than on the RGB-D set, but the input features are only based on depth values and not on color.

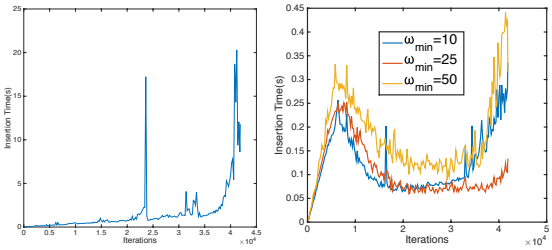


Fig. 7: Vertex insertion times for OSC (left) and our online clustering approach (right). Note that our approach never takes more than 0.5 seconds and that run time increases initially, because more new clusters are created. Later, new vertices often are satellites, which reduces the run time.

VI. Experiments

We evaluated our approach on two different data sets: the RGB-D data set provided by Lai *et al.* [20] with 41876 instances of 51 different object classes, and a subset of the KITTI data set of 3D point clouds in an urban environment [21], which contains objects from 7 classes, namely 1265 cars, 775 cyclists, 1035 Pedestrians, 957 vans, 667 trucks, 223 sitters and 257 bakground objects (misc). For both data sets, we compute HMP features, while for point clouds we first compute depth images (see Sec. III). For the HMP features, we first learn a dictionary of size 75 on the first level with K-SVD for depth and gray channels, and of size 150 for normal vectors and RGB channels. Then, on the second level we learn dictionaries of size 500 for gray scale and depth, and 1000 for RGB and normals. For each RGB-D image we compute an HMP feature vector of length 42000.

A. Similarity Analysis

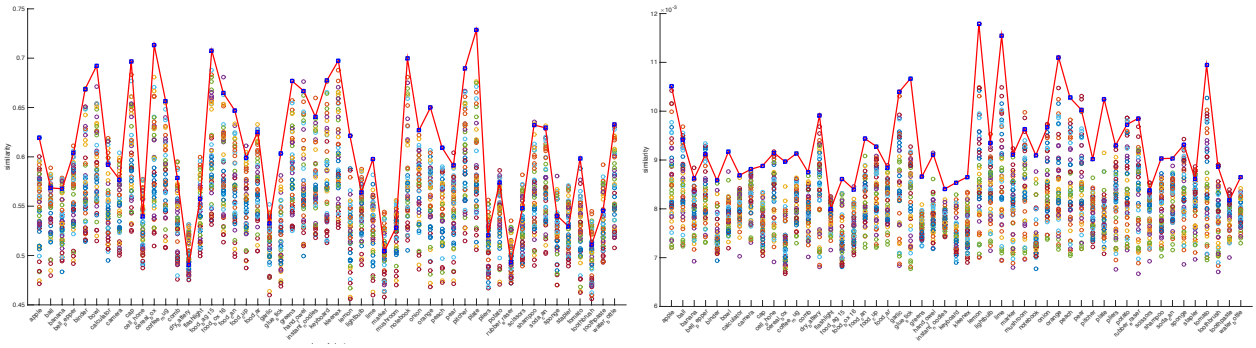
To find a good similarity measure (kernel) for our online clustering algorithm, we ran a specific test on 10,000 samples from the RGB-D data set. For each pair of images within the same object class and across different object classes we computed similarities and the corresponding average similarities. The result for the Gaussian kernel and the Inverse City Block (ICB) kernel are shown in Fig. 4. For each class, a colored circle refers to the average similarity with another class. Blue circles, which are connected with red lines depict the average self-similarity of each class. Thus, we can see that the self-similarity values tend to be better for the ICB kernel than for the Gaussian kernel. Therefore, in our following experiments, we only used the ICB kernel.

B. Online Learning of 3D Objects

To assess the performance of our approach we randomly shuffle the 41876 different cropped images from the RGB-D data set and present them to our online SSL algorithm. We compare the results with the baseline method OSC+LP, described in Sec. IV, where we use the following criteria. First, the V_1-measure [22], which is a measure for cluster quality and consists of the harmonic mean between *homogeneity* and *completeness*. Intuitively, homogeneity is closely related to purity, i.e. it is high if clusters consist of many samples from the same object class. In contrast, completeness is high if many samples from an object class are in the same cluster (for details see [22]). Our second criterion is the number of clusters divided by the total number of vertices. Ideally, this value should be low because then we have less label queries. Finally, the third criterion is the classification accuracy, i.e. the percentage of correctly classified samples.

Fig. 5 shows the results on the whole RGB-D data set, where the left part shows our results, with $\omega_{min} = 25$, and the right part the ones obtained with OSC+LP with a threshold $\vartheta = 0.009$. As we can see, our method outperforms the OSC+LP algorithm both in terms of cluster homogeneity and in final classification rate. At the same time, the number of clusters produced by our algorithm is only slighty higher. This is good, because the number of clusters is directly related to the number of label queries. This is shown in Fig. 6 (left), where we plot the accumulated number of generated label queries for both methods. We see that, compared to OSC+LP our approach only requires very few ground truth labels for learning. For the KITTI data we obtain the results of Fig. 6 (right). Note that we only use a subset of the data because the entire data set is very unbalanced between the classes. We see that the classification is worse than the one on RGB-D, but the feature vectors only contain depth values.

Furthermore, we compare our approach with OSC+LP in terms of run time needed for a vertex insertion, as this was also one of our main design goals. Fig. 7 (left) shows the



(a) Gaussian Kernel with $\sigma = 1.0$

(b) Inverse CityBlock Kernel

Fig. 4: Comparison of two similarity measures (kernels). For each object class in the RGB-D data we show the average similarity across classes as colored circles, and the self-similarity for each class (blue circles, connected with red lines).

insertion time for OSC+LP for each iteration (insertion) on the RGB-D data. We can see one major peak at around 23000 observed samples. We relate this to an extremely large amount of cluster rearrangements required at this stage. Later, towards the end of the data set, the run time increases again very quickly. In contrast, insertion in our algorithm never takes more than 0.5 seconds (see right plot in Fig. 7).

VII. Conclusions

As we have shown, the combination of semi-supervised learning methods with online clustering can be a very efficient approach for learning 3D object classes from large data streams. However, a straightforward implementation using standard Online Star Clustering and Label Propagation results in a suboptimal performance, both in run time and in accuracy, because OSC is not particularly designed for combination with SSL. In contrast, if we modify the clustering algorithm accordingly, we obtain impressive results for 3D object classification with comparably little effort in terms of run time and generated label queries. As a consequence, our proposed method is very well suited for challenging online classification tasks in mobile robotics.

Acknowledgement: The work in this paper was funded by the EU project SPENCER (ICT-2011-600877).

References

[1] J. Aslam, E. Pelekhov, and D. Rus, "The star clustering algorithm for static and dynamic information organization," *Journal of Graph Algorithms and Applications*, vol. 8, no. 1, pp. 95–129, 2004.

[2] J.-P. Zhang, F.-C. Chen, L.-X. Liu, and S.-M. Li, "Online stream clustering using density and affinity propagation algorithm," in *Software Engineering and Service Science (ICSESS)*, 2013, pp. 828–832.

[3] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. MIT Press, 2006.

[4] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. of the International Conference on Machine Learning (ICML)*, 1999, pp. 200–209.

[5] N. D. Lawrence, J. C. Platt, and M. I. Jordan, "Extensions of the informative vector machine," in *Proc. of the First Intern. Conf. on Deterministic and Statistical Methods in Machine Learning*. Springer-Verlag, 2004, pp. 56–87.

[6] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon University, Tech. Rep. CMU-CALD-02-107, 2002.

[7] W. Barbakh and C. Fyfe, "Online clustering algorithms," *International Journal of Neural Systems (IJNS)*, vol. 18, no. 3, pp. 1–10, 2008.

[8] A. Choromanska and C. Monteleoni, "Online clustering with experts," in *ICML W. on Online Trading of Exploration and Exploitation*, 2011.

[9] O. Cappé, "Online EM algorithm for hidden markov models," *J. of Comput. and Graphical Statistics*, vol. 20, no. 3, pp. 728–749, 2011.

[10] S. Yildirim, S. S. Singh, and A. Doucet, "An online expectation–maximization algorithm for changepoint models," *J. of Comput. and Graphical Statistics*, vol. 22, no. 4, pp. 906–926, 2013.

[11] R. Triebel, R. Paul, D. Rus, and P. Newman, "Parsing outdoor scenes from streamed 3d laser data using online clustering and incremental belief updates," in *Robotics Track of AAAI Conference on Artificial Intelligence*, 2012.

[12] L. Bo, X. Ren, and D. Fox, "Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms," in *Advances in Neural Information Processing Systems (NIPS)*, December 2011.

[13] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. of the International Conference on Artificial Intelligence and Statistics*, vol. 5, 2009, pp. 448–455.

[14] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[15] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. of the 25th International Conference on Machine Learning*. New York, NY, USA: ACM, 2008, pp. 1096–1103.

[16] L. Bo, X. Ren, and D. Fox, "Unsupervised Feature Learning for RGB-D Based Object Recognition," in *ISER*, June 2012.

[17] J. Maye, R. Triebel, L. Spinello, and R. Siegwart, "Bayesian online learning of driving behaviors," in *Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[18] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-Time RGB-D based People Detection and Tracking for Mobile Robots and Head-Worn Cameras," in *Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[19] Y. Bengio, O. Delalleau, and N. L. Roux, *Semi-supervised Learning*. MIT Press, 2006, ch. Label Propagation and Quadratix Criterion.

[20] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[21] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Intern. Journal of Robotics Research (IJRR)*, 2013.

[22] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 410–420.