The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Learning to Hash Logistic Regression for Fast 3D Scan Point Classification

Jens Behley, Kristian Kersting, Dirk Schulz, Volker Steinhage and Armin B. Cremers

***Abstract*— Segmenting range data into semantic categories has become a more and more active field of research in robotics. In this paper, we advocate to view this task as a problem of fast, large-scale retrieval. Intuitively, given a dataset of millions of labeled scan points and their neighborhoods, we simply search for similar points in the datasets and use the labels of the retrieved ones to predict the labels of a novel point using some local prediction model such as majority vote or logistic regression. However, actually carrying this out requires highly efficient ways of (1) storing millions of scan points in memory and (2) quickly finding similar scan points to a target scan point. In this paper, we propose to address both issues by employing Weiss *et al.*'s recent spectral hashing. It represents each item in a database by a compact binary code that is constructed so that similar items will have similar binary code words. In turn, similar neighbors have codes within a small Hamming distance of the code for the query. Then, we learn a logistic regression model locally over all points with the same binary code word. Our experiments on real world 3D scans show that the resulting approach, called spectrally hashed logistic regression, can be ultra fast at prediction time and outperforms state-of-the art approaches such as logistic regression and nearest neighbor.**

I. Introduction

Classification of sensor data is a fundamental ability needed by autonomous robots operating in natural and changing environments. It enables the systems to distinguish properties of objects in their surrounding, which first of all, is necessary for identifying basic characteristics like the drivability of terrain or the compliance of obstacles. Classification can further more act as a filter to detect useful features in the data, for example to improve self-localization and map building approaches. Above all, classification is often a prerequisite to detect task-relevant objects, which is a cornerstone for every high-level behavior of an intelligent autonomous system.

For these reasons, the classification of sensor readings, both camera images and laser scan data, into semantically meaningful classes has received a lot of attention in computer vision and robotics. Especially in the robotics community, the interpretation of 3D laser scans has become an active field of research, and very good results have been achieved recently [1], [2], [3], [4], [5]. However, most of the proposed approaches use computationally expensive statistical inference techniques for the classification, but modern 3D laser sensing devices easily produce massive datasets of up to now 1.3 million laser points per second. The current techniques are, therefore, not able to classify the data on-line. In this article, we address this problem and propose a new fast approach based on novel ideas from machine learning for the classification of mass data.

A recent exciting development in the machine learning community has been the insight that massive datasets are not only challenging but can also be viewed as an opportunity [6]. Machine learning and data mining techniques typically consist of two parts: the model and the data. Most effort in recent years has gone into the modeling part. Massive datasets, however, allow one to move into the opposite direction: how much can the data itself help us to solve the problem? Halevy *et al.* [7] even speak of "the unreasonable effectiveness of data". Massive datasets are likely to capture even very rare aspects of the problem at hand. Does this also hold for 3D scan point classification tasks? Can we learn the characteristics of objects from very dense laser points without learning complex models? These are exactly the problems we examine in this paper.

Specifically, we investigate the question of classifying objects within 3D scans. This is a challenging problem as it is a highly non-linear optimization task. Consider for instance detecting cars in 3D laser scans. A car is not just a single surface but it is composed of flat and curved surfaces. In turn, it is difficult – if not impossible – to elegantly describe them in terms of simple geometrical features so that the feature vectors form a linear separable cluster in feature space. It is more likely that their descriptions scatter in the feature space. Indeed, we may overcome this problem by using *collective classification* approaches. They take the surrounding of a laser point into account: intuitively, class labels should propagate smoothly among neighboring points. The increased performance, however, comes at the expense of much higher computational costs for learning and inference. When very many scans are available, simple scan indexing techniques can be used to retrieve scans with object arrangements similar to the query scan. If we have a big enough database then we can find, with high probability, scans looking very close to a query scan, containing similar scenes with similar objects arranged in similar spatial configurations. Moreover, if scans in the retrieval set are (partially) labeled, then we can propagate the labels to the query scan and in turn

J. Behley , V. Steinhage and A. B. Cremers are with the Department of Computer Science III, University of Bonn, 53117 Bonn, Germany.
K. Kersting is with the Knowledge Discovery Department, Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany.
D. Schulz is with the Unmanned Systems Department, Fraunhofer FKIE, 53343 Bonn-Wachtberg, Germany.
Kristian Kersting was supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.
{behley, steinhage, abc}@iai.uni-bonn.de,
kristian.kersting@iais.fraunhofer.de,
dirk.schulz@fkie.fraunhofer.de

perform classification using some simple local models such as majority vote or logistic regressions.

Although conceptually simple, actually carrying out nearest neighbor approaches requires highly efficient ways of (1) storing millions of scans in memory and (2) quickly finding similar scans to a target scan. Our main contribution is to address both issues by representing each item in a database by a compact binary code that is constructed so that similar items will have similar binary code words. In turn, similar neighbors have codes within a small Hamming distance of the code for the query. Then, we learn a logistic regression model locally over all points with the same binary code word. More precisely, we use Weiss *et al.*'s recent spectral hashing to compute the compact binary codes [8]. Using codes learned by spectral hashing, retrieval can be amazingly fast – millions of queries per second on off-the-shelf PCs. Our experiments show that the resulting approach, called spectrally hashed logistic regression, obeys a constant time complexity for classification of 3D laser points. Moreover, the fast classification performance does not sacrifice accuracy. Spectrally hashed logistic regression works surprisingly well in our application: identifying cars, foliage, walls and load bearing areas in 3D laser scans. The laser scans are produced by a Velodyne laser scanner mounted on a mobile robot that is equipped with an inertial navigation system (INS). The position and rotation information from the INS allows us to register the laser scans approximately. To our knowledge, we are the first to apply spectral hashing to a robotics task and in combination with logistic regression.

We now proceed as follows. First, we discuss related work in section II. In section III, we introduce the proposed classification approach using spectral hashing, which is also briefly introduced in this section. Section IV presents our current results and finally section V concludes and outlines future work.

II. Related Work

In robotics mostly data-driven approaches based on Conditional Random Fields (CRF) [9] have been used to classify 3D point clouds. Angulov *et al.* [1] introduced Associative Markov Networks [10] for this purpose, and most of the up-following approaches were based on this *collective classification* approach. However, these techniques require quadratic programming and linear programming, for learning and inference respectively, which is almost intractable for larger point sets. Several methods have been proposed to speed up the process, either by using data reduction [2], [3] or by more efficient learning and inference methods [11], [4], [5]. In the following, we will briefly mention the most recent approaches for supervised 3D laser scan classification and summarize their main ideas.

Munoz *et al.* showed in [4], how high-order interactions between cliques instead of pair-wise couplings and already classified scans can be used to allow accurate on-board classification. Furthermore, they proposed to use functional gradient boosting [12] for learning node potentials as weighted sums of linear regressors instead of the usually used log-linear potentials [5] . Agrawal *et al.* [13] augmented a CRF with object potentials generated by segmenting the scene into objects and calculating the covariances of the objects' laser points. Lai and Fox [14] applied an exemplar approach using 3d models from the web, and employed domain adaption in order to remove artefacts not visible in real laser scans. Patterson *et al.* [15] employed a nearest neighbor approach using spin images [16] and extended Gaussian images (EGI) [17]. First a set of reference points is sampled from the labeled training scene and spin images are computed. The spin images are stored in a database. When classifying unseen scans spin images of reference points are matched against the database, and clustered hypotheses verified using the EGIs.

In general, a lot of effort has been invested into more complex models and most of the approaches need a lot of processing power to classify laser points. As we pointed out in the introduction, we are moving in the opposite direction, inspired by the work of Torralba *et al.* [18] who employed the power of a vast number of images to label arbitrary scenes according to a very, very large database of images from the well-known LabelMe dataset. They used distance-preserving hashing to enable a fast retrieval of approximate nearest neighbors.

Finally, we have to mention the well-known locally weighted learning of Atkeson *et al.* [19], which learns a local model for every query point using k nearest neighbors from the training data. These neighbors are weighted according to the distance to the query point. Now, our aim is to avoid the need for the exact calculation of the k nearest neighbors, as this is too expensive for larger sets of training data.

III. Spectrally Hashed Logistic Regression

Assume that we have a huge amount of scan (points), say N, and that the decision boundaries are very irregular. In this case, nearest neighbor approaches are an elegant and very flexible tool for classification. However, having fast techniques for finding nearest neighbors to a query is then essential.

Recently, hashing methods for fast retrieval have received a lot of attention within the machine learning community, see e.g. [8], [20]. They learn a mapping from the input data to a low-dimensional Hamming, i.e., binary space. Note that the fact that the embeddings are binary is critical to ensure fast retrieval times. As [8], [20] report, this kind of retrieval can be amazingly fast; millions of queries per second on standard computers. This is because the Hamming distance between two objects can be computed via an xor operation and a bit count. Moreover, if the input dimensionality is very high, as in our case, hashing methods lead to enormous computational savings as few bits are often already sufficient to encode compactly the whole dataset.

Hashing naturally leads to the following 3D scan points classification approach:

1) **(Hashing)** Learn a compact binary code for a given set of N scans.

2) **(Local Classification)** Learn a local classification model such as majority vote or logistic regression on all scans that have the same binary code.
3) **(Prediction)** For classifying a new scan (point) x, compute the code of x, look-up the associated local model, and use it to assign a class label to x.

Indeed, this non-parametric large-scale classification approach is a special case of locally weighted regression performing classification around a point of interest using all training scans that have identical binary codes only. As we will argue in the next section, if the look-up of the code for a new scan is done in a clever way, this can yield ultra fast classification performance. Furthermore, as our experimental evaluation will show, this approximation works surprisingly well in our classification setting. It actually outperforms nearest neighbor and logistic regression.

A. Spectral Hashing

For computing the compact binary codes, we are using Weiss *et al.* spectral hashing [8]. The main benefit of spectral hashing is that the partitioning of the feature space can be computed in linear time.

Spectral hashing works as follows. To preserve distances, one is interested in a hash function that maps nearby data points x_i and x_j to binary hash codes that have a small Hamming distance. Thus, the objective for a hash function $h : \mathbb{R}^n \mapsto \{0,1\}^k$, which helps us to search efficiently in large datasets $x_i \in \mathbb{R}^n$ that is distributed according to a distribution $p(x)$, can be formulated as follows:

$$\text{min.} \int K(x_i, x_j) \cdot ||h(x_i) - h(x_j)||^2 \cdot p(x_i) \cdot p(x_j) \, dx_i \, dx_j \tag{1}$$

$$\text{s. t.} \int h(x)p(x) \, dx = 0 \tag{2}$$

$$\int h(x)h(x)^T p(x) \, dx = \mathbf{Id} \tag{3}$$

Here, the function $K(x_i, x_j)$ defines the similarity between different data points. A common choice is the Gaussian kernel $K(x_i, x_j) = exp(-||x_i - x_j||^2/\epsilon^2)$. The two constraints encode the requirement that the different bits of hash codes should be independent (Eq. 2) and uncorrelated (Eq. 3). As Weiss *et al.* [8] have shown, finding such codes is NP hard. To overcome this problem, they relax the constraint that the codewords need to be binary, $h(x) \in \{0,1\}^k$. This relaxed problem can be solved in polynomial time. Indeed, it has been shown that the solution is given by *eigenfunctions* $\Phi(x)$. If $p(x), x = \left(x^{(1)}, x^{(2)}, \ldots, x^{(n)}\right) \in \mathbb{R}^n$ is separable, i.e. $p(x) = \prod_i p_i(x^{(i)})$, and the similarity is defined by the Gaussian kernel then the solution $\Phi(x)$ is given by the product of the 1-dimensional eigenfunctions

$$\Phi_1\left(x^{(1)}\right)\Phi_2\left(x^{(2)}\right)\cdots\Phi_n\left(x^{(n)}\right) \tag{4}$$

and eigenvalue $\lambda_1 \cdot \lambda_2 \cdots \lambda_n$. Especially, if $p(x)$ is a uniform distribution on $[a,b]$, the eigenfunctions $\Phi_k(x)$ are given by

$$\Phi_k(x) = sin\left(\frac{\pi}{2} + \frac{k\pi}{b-a}x\right) \tag{5}$$

$$\lambda_k = 1 - exp\left(-\frac{\epsilon^2}{2}\left|\frac{k\pi}{b-a}\right|^2\right). \tag{6}$$

Assuming that data is uniformly distributed, we can now calculate the eigenfunctions and threshold the values at 0 to obtain a codeword. This results in the following algorithm to determine a hash function h for data points $\mathcal{X} = \{x_i \in \mathbb{R}^n\}$:

1) Calculate the k principle components using eigenvalue decomposition of the covariance matrix. Rotate the data x_i according to the k largest eigenvectors, resulting in $\tilde{x}_i^{(j)}, 0 \le j < k$.
2) Determine for every dimension $a^{(j)} = min_j\left(\tilde{x}_i^{(j)}\right)$ and $b^{(j)} = max_j\left(\tilde{x}_i^{(j)}\right)$ and evaluate the eigenvalues according to (6). Sort the eigenvalues to find the k smallest eigenvalues.
3) Threshold the resulting eigenfunctions $\Phi_k(x)$ with smallest eigenvalue at 0, to obtain the hash code.

As shown in [8], the algorithm is not restricted to uniformly distributed data, and can generate hash codes that are capable to find a good partition of the data, which allows to search efficiently for nearest neighbors. We will show in the next section, how the space of data points is partitioned using this hashing algorithm. Furthermore, we will show that the calculation of the hash function can be done efficiently, since we do not need to handle every data point explicitly: computing the covariance is sufficient. In turn, we only have to determine the minimum and maximum of the rotated feature vectors to get an partition of the feature space.

B. Combining Spectral Hashing and Logistic Regression

The main idea underlying locally weighted regression is to use local models of linear regression learned from k neighboring points of a query point. By using this lazy classification, one could approximate even non-linear target functions with local linear regression models. However, finding the k nearest neighbors can get rather complex in high-dimensional data, so that the computational cost grows with an increasing set of training points.

To overcome this, we partition the feature space using spectral hashing, learn local models directly from the training data and store these local logistic models for every partition given by the hash function h, when necessary. If only feature vectors from one class y lie in a hash bin, we just store a bias weight β in the weight vector w_y and the rest of the weight vectors is initialized with $-\beta$ as bias weight. This is summarized in algorithm 1. More precisely, we use a binary logistic regression with L2-regularization and train it in one-against-all fashion to get a multi-class classification. But indeed the algorithm is not restricted to logistic regression models, so that even non-linear classifiers could be used to classify locally within a partition defined by the hash function. To get the class of an unseen laser point, we just

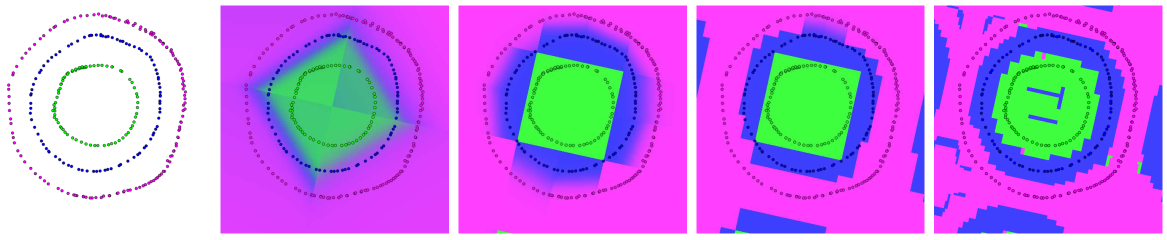

(a) (b) (c) (d) (e)

Fig. 1. Some examples of the partition of a highly non-linear feature space (a) using the proposed spectrally hashed logistic regression. Subfigure (b) is generated using 2 bits, (c) uses 4 bits, (d) was trained with 8 bits and in (e) we used 16 bits for hashing of the feature space. The repetition in class assignments is caused by the sinusoid in the eigenfunction in conjunction with a dimension, where the minimum and maximum of the feature vectors in that dimension is not the same as the minimal and maximal value of that dimension. (Best viewed in color.)

compute the binary code using the learned hash function and retrieve the right local logistic model for classification, the one with the same binary code. If no logistic model is associated with that code, we compute the mean predictions of all nearest models in the same Hamming distance.

Figure 1 visualizes some examples of spectrally hashed logistic regression for different number of bits used for the hash codes. As on can see, with increasing hash size, the partitioning increases and also the decision boundaries of the local logistic regression models adapt to the non-linear feature space, as we have argued in the beginning of this section. Furthermore, smaller partitions lead to less data points inside a partition, thus the learning of the logistic regression can be performed more efficiently due to the reduced size of the training set. But also a negative side effect is observable: as the number of possible bits increases, it gets more likely to perform overfitting, as we will see in the next section.

Algorithm 1: Learn local logistic regression models

```
Data: training set $\mathcal{X} = \{(x_i, y_i)\}$
with features $x_i \in \mathbb{R}^d$ and labels $y_i \in \mathcal{Y}, |\mathcal{Y}| = K$
Result: hash function $h$, local logistic models
      $\mathbf{L_i} = \{(w_0, \ldots, w_K)\}$
learn hashing function $h$ (cf. section III-A)
/* build hashtable                                    */
foreach $(x_i, y_i) \in \mathcal{X}$ do
    $h_i := h(x_i)$
    $H[h_i] := H[h_i] \cup (x_i, y_i)$
end
/* learn local models $L_i$                            */
foreach $h_i \in [0, 2^{k-1}] \wedge H[h_i] \neq \emptyset$ do
    if $|\{y|(x,y) \in H[h_i]\}| = 1$ then
        foreach $y \in \mathcal{Y}$ do
            $w_j = \begin{cases} (\beta, 0, \cdots, 0) & , j = y \\ (-\beta, 0, \cdots, 0) & , otherwise. \end{cases}$
            store $w_j$ in local models $L_i$.
        end
    else
        foreach $y \in \mathcal{Y}$ do
            learn logistic regression with weights $w_y$.
            store $w_y$ in local models $L_i$.
        end
    end
end
```

IV. Evaluation

To evaluate our approach, we use a dataset recorded with a Velodyne 3D laser scanner mounted on an QinetiQ Longcross platform. The robot is equipped with an Oxford Ltd. inertial navigation system (INS) that is sufficiently precise to allow scan registration without additional mapping software. As mentioned before, the laser scanner produces 1.3 million laser points per second. As we want to compare different classification approaches, we down-sampled single $360°$ laser scans from approx. 86.000 to approx. 8.600 by taking only every tenth measurement from the scanner. By reducing the laser scans in this manner, we end up by 129.000 laser points per second.

All experiments were done using 10-fold stratified cross-validation and the classification accuracy is averaged over all folds. The experiments were performed using precomputed feature vectors, so that the timing results do not include the time required for the evaluation of the features. The training set has been randomized before doing cross-validation. However, we used for every cross-validation the same random seed to get comparable results. We used a uniform class distribution for learning the classifiers. This reduces the influence from classes that are significantly more prominent, such as load bearing areas or foliage.

The *velodyne dataset* has been labeled manually with different classes: *vehicle*, *ground*, *building* and *vegetation*. The class *ground* consists drivable area, but also sidewalks and lawn. The class *vegetation* is quite diverse and contains all kinds of plants as shrubs, foliage, trees and bushes. Overall 3.371.808 points has been labeled in which 2.419.584 (71.7%) points are *ground*, 295.723 (8.8%) points are *vehicle*, 38.326(1.1%) points are *building* and 618.175(18.3%) points are *vegetation*.

A. Features

In the experiments, we tried several features and feature combinations proposed in different prior approaches, and the well-known spin images [16] showed the best results. However, we also try to detect walls and buildings, and spin

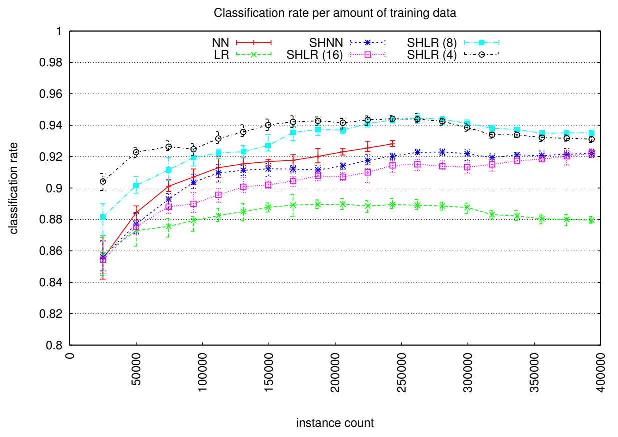


Fig. 2. Classification rates using 10 fold cross-validation with different classification approaches.

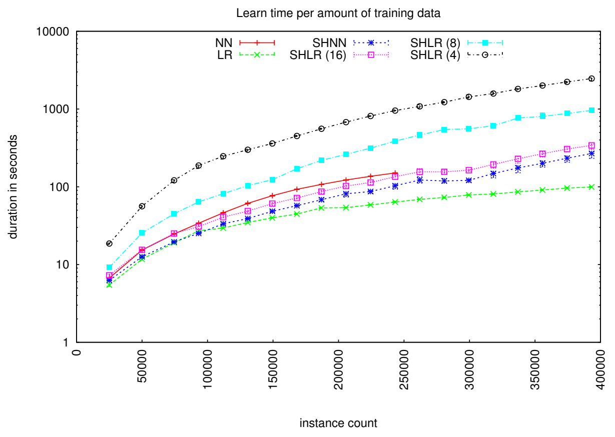


Fig. 3. Time needed for learning different classification approaches.

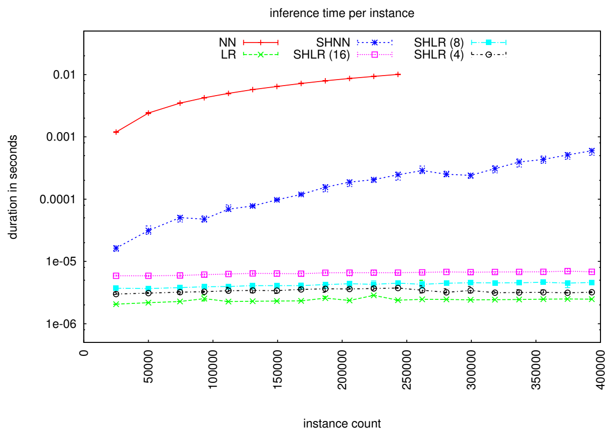


Fig. 4. Time needed for classification of one laser point (without computation of the features).

images in the original implementation, are not suitable to distinguish between those, because they achieve rotational invariance of features by using the point's normal as spin-axis. We compensate this shortcoming by using the up-vector instead, as proposed by Agrarwal *et al.* [13] for their local shape histograms. The up-vector is determined by the orientation information of the INS system employed. We used in all experiments a bin size of 0.1 m and 20 bins per dimension.

B. Results

To test our approach, we compared the results with different other local classifiers. First, we used a binary logistic regression (LR)[1] [21] as baseline with the same amount of training data and extended it to a multi-class classifier by training it in one-against-all fashion. Besides this, we also implemented a nearest neighbor classifier (NN) using the ANN library [22]. However, due to the high computational costs of querying nearest neighbors in high-dimensional data, we only evaluated this approach up to 250.000 data points. All classifiers were implemented in C++ and the experiments were performed on an Intel Xeon X5550 with 2.67 GHz using a single core and 12 GB memory.

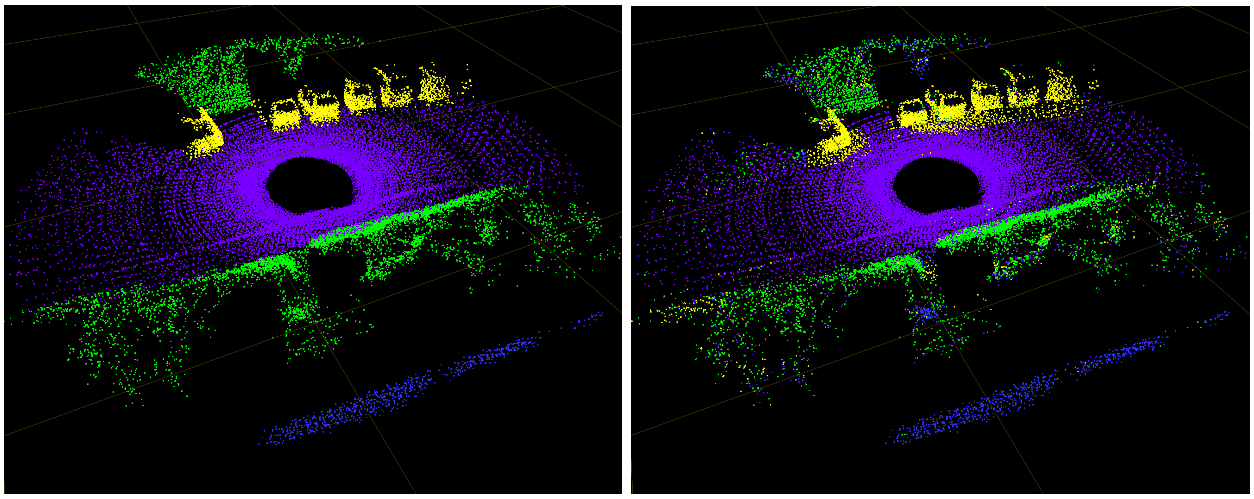
Because we are also interested in the performance of spectral hashing as nearest neighbor approach (SHNN), we implemented a nearest neighbor classifier using the learned hashing function. Like in algorithm 1, a hash table is generated and the feature vectors hashed to the same code are stored in a linked list for that bin. In experiments the usage of 16 bits and a Hamming distance of 1 to search for the exact nearest neighbor yielded the best results.

Fig. 2 depicts the classification rates achieved. The nearest neighbor using the kd-tree easily outperforms logistic regression, which confirms our intuition that the feature space is not linearly separable and nearest neighbor approaches can handle this. Also the nearest neighbor using the hash function of spectral hashing outperforms the logistic regression. Notable is the time required for classification using the hash compared to the kd-tree as depicted in Fig. 4. Using hashing results in a significant decrease of search time for nearest neighbors, and the classification results remain comparable.

Nevertheless, our approach with 8 bits and 4 bits outperforms even the nearest neighbor approach. This can be explained by the usage of logistic regression, which is less sensitive to outliers. The classifier with 16 bits performed worse than the others using less bits due to over-fitting using insufficient training data. Fig. 5 shows an example laser scan with a point classification achieved by our method. The approach produces very good results, but still has some difficulties to distinguish between the classes foliage and cars, which results from very similar appearance of some lower bushes with the front part of cars. Also notable is the decrease of classification accuracy, in regions where the scan gets sparser.

The even more interesting aspect of our approach is the inference time (see Fig. 4) and the time needed for learning (see Fig. 3). In Fig. 4 logistic regression and the proposed spectrally hashed logistic regression are the most efficient classifiers, since in the case of logistic regression only the weight vector has to be applied to the feature vector and scaled by the sigmoid function. In our approach the

[1]Publicly available at http://www.autonlab.org.



(a) (b)

Fig. 5. Classification result of our proposed approach (b) compared to a (a) manual labeling. The image is composed of 5 360 degree scans which has been classified separately. The classes are color-coded as follows load bearing = purple, car = yellow, vegetation = green and walls = blue. (Best viewed in color.)

calculation is a little bit more time consuming, since we have to rotate the data and evaluate the hash function. Learning is more time consuming than logistic regression, especially when using only a small number of bits.

V. Conclusions and Future Works

We have presented an extremely simple and hence efficient algorithm for classifying 3D scan points called spectrally hashed logistic regression. As shown, one simply learns a hash function using spectral hashing – perform PCA on the data and fit a multidimensional rectangle; the aspect ratio of this multidimensional rectangle determines the code using a simple formula – and uses it to look-up local logistic regression models that are learned on scans mapped onto the same code. Despite its simplicity, its performance is superior to state-of-the-art methods in our experiments.

There are several interesting avenues for future work. Next to running more experiments, one should investigate other (combinations of) features to improve the classification. Another promising avenue is the overall speed-up of the calculation of the features, which is currently the bottleneck in our current implementation. Finally, it is interesting to start exploring what can be called hashed locally learning in general, i.e., easy-to-implement classification or regression approaches that easily scale to millions of data items and run at real-time. Our experimental results are an encouraging sign that this may not be insurmountable.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, “Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data,” in *CVPR*, vol. 2, 2005, pp. 169–176.

[2] R. Triebel, K. Kersting, and W. Burgard, “Robust 3D Scan Point Classification using Associative Markov Networks,” in *ICRA*, 2006, pp. 2603–2608.

[3] E. H. Lim and D. Suter, “Conditional Random Field for 3D point clouds with Adaptive Data Reduction,” in *Int. Conf. on Cyberworlds*, 2007, pp. 404–408.

[4] D. Munoz, N. Vandapel, and M. Hebert, “Onboard Contextual Classification of 3-D Point Clouds with Learned High-order Markov Random Fields,” in *ICRA*, 2009, pp. 4273–4280.

[5] D. Munoz, J. A. D. Bagnell, N. Vandapel, and M. Hebert, “Contextual Classification with Functional Max-Margin Markov Networks,” in *CVPR*, 2009, pp. 975–982.

[6] A. Torralba, R. Fergus, and W. T. Freeman, “80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition,” *TPAMI*, vol. 30, no. 11, pp. 1958–1970, 2008.

[7] A. Halevy, P. Norvig, and F. Pereira, “The Unreasonable Effectiveness of Data,” *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[8] Y. Weiss, A. Torralba, and R. Fergus, “Spectral Hashing,” in *NIPS*, 2009, pp. 1753–1760.

[9] J. Lafferty, A. McCallum, and F. Pereira, “Conditional Random Fields: Probabilstic Models for Segmenting and Labeling Sequence Data,” in *ICML*, 2001, pp. 282–289.

[10] B. Taskar, V. Chatalbashev, and D. Koller, “Learning Associative Markov Networks,” in *ICML*, 2004, pp. 807–814.

[11] D. Munoz, N. Vandapel, and M. Hebert, “Directional Associative Markov Network for 3-D Point Cloud Classification,” in *Proc. of the* 4^{th} *Int. Symp. on 3D Data Processing, Visualization and Transmission*, 2008, pp. 63–70.

[12] N. Ratliff, J. A. Bagnell, and S. Srinivasa, “Imitation Learning for Locomotion and Manipulation,” in *Humanoids*, 2007.

[13] A. Agrawal, A. Nakazawa, and H. Takemura, “MMM-classification of 3D Range Data,” in *ICRA*, 2009, pp. 2269–2274.

[14] K. Lai and D. Fox, “3D laser scan classification using web data and domain adaptation,” in *RSS*, 2009.

[15] A. Patterson, P. Mordohai, and K. Daniilidis, “Object Detection from Large-Scale 3D Datasets using Bottom-up and Top-down Descriptors,” in *ECCV*, 2008, pp. 553–566.

[16] A. Johnson and M. Hebert, “Using spin images for effcient object recognition in cluttered 3D scenes,” *TPAMI*, vol. 21, no. 5, pp. 433–449, 1999.

[17] B. Horn, “Extended gaussian images,” *Proc. of the IEEE*, vol. 72, no. 12, pp. 1656–1678, 1984.

[18] A. Torralba, R. Fergus, and Y. Weiss, “Small codes and large databases for recognition,” in *CVPR*, 2008.

[19] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally Weighted Learning,” *AI Review*, vol. 11, pp. 11–73.

[20] R. Salakhutdinov and G. Hinton, “Semantic Hashing,” *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[21] P. Komarek and A. Moore, “Making Logistic Regression A Core Data Mining Tool: A Practical Investigation of Accuracy, Speed, and Simplicity,” technical report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep., May 2005.

[22] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, “An optimal algorithm for approximate nearest neighbor searching fixed dimensions,” *J. of the ACM*, vol. 45, pp. 891–923, 1998.