Object Recognition in 3D Lidar Data with Recurrent Neural Network

Danil V. Prokhorov
Toyota Research Institute NA
TTC-TEMA, Ann Arbor, MI 48105, USA
dvprokhorov@gmail.com

Abstract

This paper introduces a new method for object recognition which is based on a recurrent neural network trained in a supervised mode. The RNN inputs 3-dimensional laser scanner data sequentially, in a natural temporal order in which the laser returns arrive to the scanner. The method is illustrated on a two-class problem with real data.

1. Introduction

A lot of literature is available on object recognition technology; see, e.g., [1], [2]. Many advances have been made to both sensing (hardware) and understanding (algorithm) technologies. The majority of them pertains to object recognition from camera images.

Typical current technology for object recognition uses images from a camera or another suitable sensor. Each image serves as an input to an object recognition algorithm. The image is usually fed into the algorithm as a collection of features, e.g., pixel intensities. The temporal order of such features is meaningless in the context of the single image. More importantly, the number of features can be very large, making the task of object recognition computationally demanding. Furthermore, reliable rotational invariant recognition of objects is an unsolved problem [3].

Instead of conventional images as inputs to an object recognition algorithm, data in the form of distance measurements from a laser scanner (lidar) is utilized directly. Traditional 2-dimensional (2D) scanners are well known as line scanners [4]. They are moved in the third dimension in order to obtain 3-dimensional (3D) information about objects. To recognize objects in such 3D world, so-called spin images have been proposed [5]. The spin image method effectively converts the object's original representation as a point cloud of laser returns into multiple-view planar projections to be classified via template matching. Another method using surface matching has been proposed in [6].

Substantially less work on object recognition has been done with lidars providing more than planar distance measurements. The paper [7] discusses airborne laser scanning technology for identifying moving objects on roads. The error rate of about 80% is achieved. Another paper [8] discusses various approaches to object recognition with lidar without their quantitative comparison. Yet another paper [9] discusses how to build obstacle maps from four simultaneous planes of otherwise standard 2D lidar.

Recently developed 3D scanner provides highly accurate distance measurements in the spherical coordinate system [10]. The 3D laser design employs many laser sending-receiving heads rotating around a vertical axis. The distance measurements can be ordered temporally according to their rotational order, instead of turning them into more conventional (planar) images as in the spin image method. As a result, temporal sequences of laser measurements for different classes of objects (e.g., a building, a vehicle, a tree, a human) will be different from each other. To distinguish "signatures" of different objects, we[1] propose to use a modest-size recurrent neural network (RNN) designed (trained) according to the description below. RNNs as powerful adaptive dynamic systems are known to be effective at recognizing potentially subtle differences between temporal sequences. We demonstrate preliminary results in recognizing different classes of objects using the RNN. Furthermore, we demonstrate how to train the RNN for mitigating rotational sensitivity of its performance to object viewing angles.

The goal of this paper is to introduce the new principle for turning the object recognition problem into the problem of recognizing temporal "signatures" of different objects originally represented as point clouds of laser returns. The method based on this principle employs RNN for processing coarsely segmented data. The method is illustrated on a two-class problem of differentiating between vehicles and non-vehicles.

It should be noted that the 3D lidar used in this paper was first used by several teams for their autonomous robotic vehicles participating in the DARPA Urban Challenge (DUC)

[1] I use plural "we" to adhere to the usage recommended for writing academic papers; see a relevant discussion on style in [12].

978-1-4244-3993-5/09/$25.00 ©2009 IEEE

2007 [11]. To the best of author's knowledge, none of the teams actually used the lidars to *classify* objects, and all used them merely to create obstacle maps of quality improved over the map quality described in [9].

We first describe the proposed method in Section 2, illustrating example views of the 3D data and performance of the RNN. We then briefly describe the current training method (Section 2.1) and the approach employed to mitigate sensitivity of RNN performance to data rotations (Section 2.2). We present our results in Section 3, followed by conclusions and future work.

2. Method

The data from a 3D scanner arrives in its natural temporal order, forming a multi-dimensional (three-dimensional) time sequence. The data stream is assumed to have three spatial dimensions: X coordinate, Y coordinate and Z coordinate of the Cartesian coordinate system (in principle, there is no limitation in terms of specific coordinate system, such as the scanners original spherical system); Z is the vertical axis. The RNN is fed by these three coordinates as inputs at every time step t of the time sequence. Figure 1 illustrates how the laser data is turned into temporal sequence. The RNN output is the class label estimated at every t. Without the loss of generality, we describe here the two-class problem, requiring just one classification output of the RNN with the classification threshold set at zero.

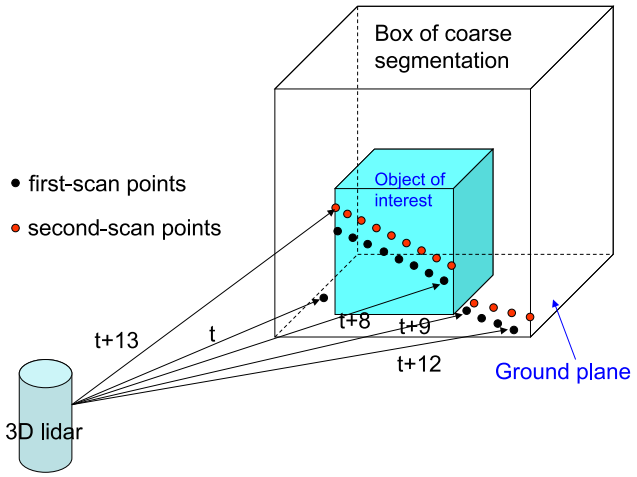

Figure 1. 3D laser scanning inside an approximate segmentation box in which classification of a target object is desired. Two example planar scans are shown (black and red), with laser return points following their temporal order (from left to right). The first point corresponds to time t, and it is located on the ground to the left of the object. The last point of the first scan corresponds to time $t+12$, followed by the first point of the second scan at $t+13$, etc.

Figures 2 and 3 illustrate how two different classes of objects (vehicle and non-vehicle) may appear when plotted in (X,Y,Z) coordinates normalized to be approximately zero mean and unity standard deviation. These pictures show approximately 70,000 to 80,000 data points for each object's time series.

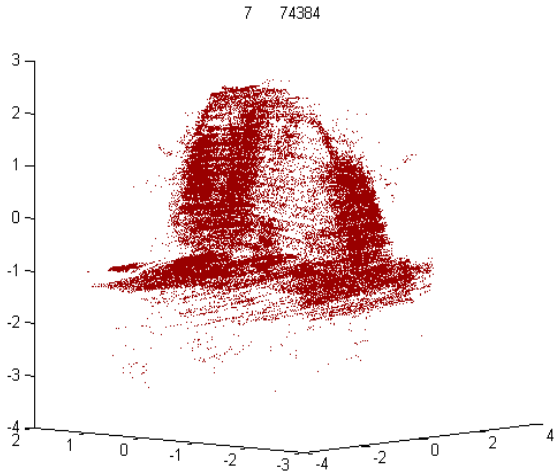

Figure 2. Example of a vehicle as "seen" by the 3D lidar (normalized view). The desired output of the RNN is set to +1 for all points.

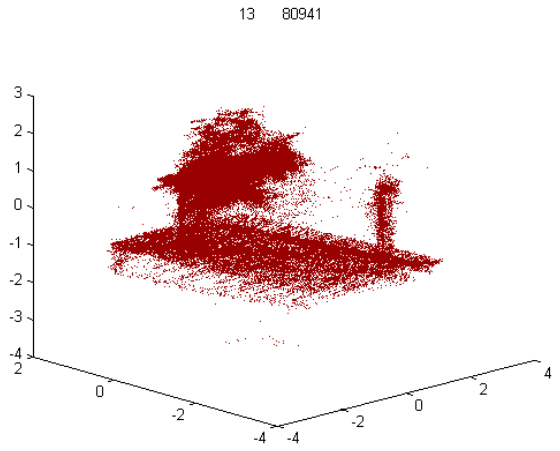

Figure 3. A non-vehicle example as "seen" by the 3D lidar (normalized view). The desired output of the RNN is set to -1 for all points.

Figures 4 and 5 show two different sets of the time sequences (for non-vehicle and vehicle) and the corresponding output sequence of the RNN (note that the RNN has been trained extensively in the process described in Section 2.1 in order to produce such an output sequence).

2.1. RNN training

We employ the standard two-hidden-layer Recurrent Multilayer Perceptron (RMLP) as our RNN. This architec-

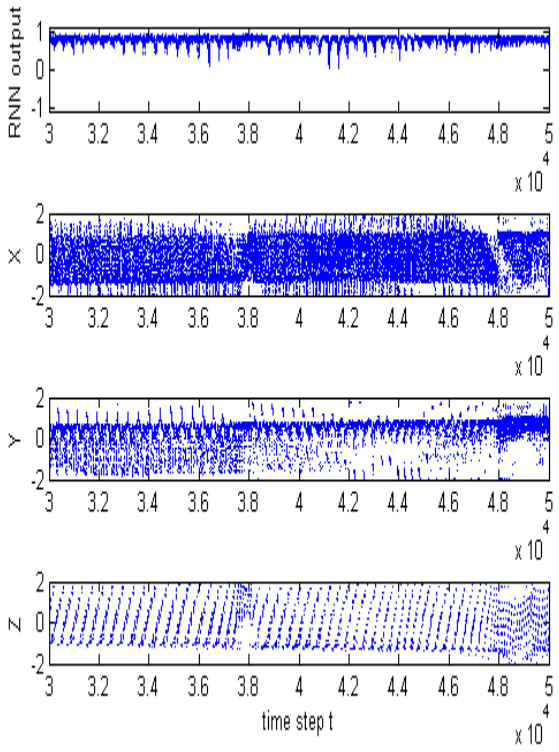

Figure 4. A fragment of the times series (time steps 30,000 to 50,000) for three coordinates X, Y and Z, along with the corresponding RNN output (top panel). All curves are represented by dots for better visibility. The RNN output stays above zero, correctly indicating that it is the vehicle shown in Figure 2.

ture is differentiable, so we can use the well known truncated Backpropagation Through Time (BPTT(h)) to compute temporal derivatives of the network's only output with respect to its weights. The RMLP weight update is currently based on the Extended Kalman Filter (EKF) method extensively described in [13], [14]. For more efficient training, we also employ training of multiple copies of the RMLP with different states (values of the recurrent nodes) but the same weights. We list steps of our training procedure below:

1. Assign the copy s of the RMLP to different time series segments (segments are randomly chosen from time series representing different objects); $s = 1, 2, ..., S$, where S is the number of copies or segments.

2. Initialize all recurrent nodes of all copies of the RMLP to zero values: $\mathbf{x}_s(0) = \mathbf{0}$.

3. Run the RMLP copies forward from time step t_0 to step $t = t_0 + H_p$, where H_p is the priming length (this length allows the RNN to develop its recurrent nodes from their initial values).

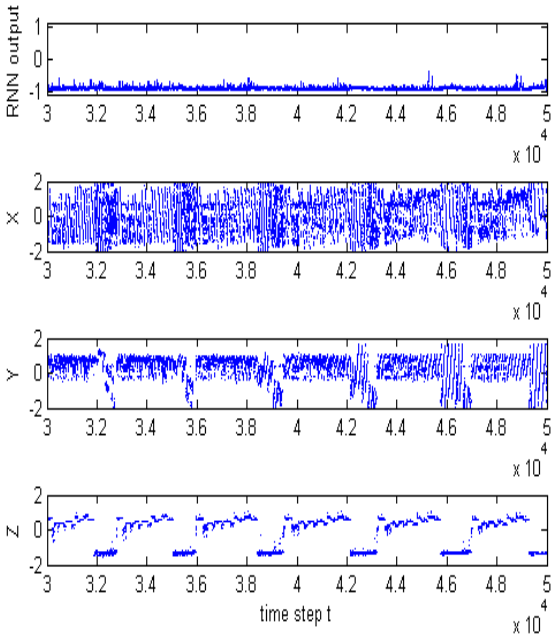

Figure 5. A fragment of the times series (time steps 30,000 to 50,000) for three coordinates X, Y and Z, along with the corresponding RNN output (top panel). The RNN output stays below zero, correctly indicating that it is the non-vehicle shown in Figure 3.

4. Run the RMLP copies forward from time step $t_0 + H_p$ to step $t = t_0 + H_p + 1$ and compute the errors for the last step t for all S segments.

5. Compute temporal derivatives of the outputs of the RMLP copies with respect to the shared weights by backpropagating to $t_0 + H_p - h$ for all S segments.

6. Adjust the shared weights according to the chosen weight algorithm using the derivatives obtained in step 5.

7. Increment t_0 and repeat the procedure beginning from step 4, etc., until the end of all segments ($t = T$) is reached.

8. One training epoch is now completed. Generate a new set of time series segments and resume training from step 1 until a sufficiently low level of the root mean square error (RMSE) is reached. (The validation set approach could also be used for stopping the training.)

Other weight update algorithms could also be used, provided that they preserve the temporal order of the time series data.

2.2. Mitigating rotational sensitivity of RNN performance

The object recognition is known to be especially difficult if the object position and orientation (pose) is not constrained, i.e., the objects may appear from an arbitrary view-

ing angle, as is the case in this object recognition problem.

We propose to equip the same RNN with a special input preprocessing in order to make the RNN less sensitive to the object viewing angle. As shown in Figure 6, the RNN receives the inputs Xrot, Yrot and Zrot after the original inputs have been multiplied by the rotational matrix. For two-dimensional rotation around Z axis implemented in this paper, X, Y are multiplied by a matrix A:

$$A = \begin{bmatrix} cos(a) & sin(a) \\ -sin(a) & cos(a) \end{bmatrix} \quad (1)$$

where a is the angle of rotation, before becoming the inputs to the RNN.

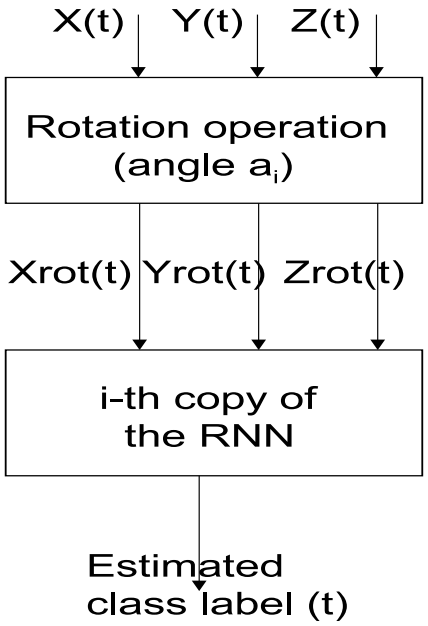

Figure 6. Structure for training to reduce RNN sensitivity to rotation of the viewpoint.

To obtain a RNN with decreased sensitivity to rotational invariance ("rotation invariant" RNN), we propose to train multiple copies of the same RNN shown in Figure 6 simultaneously on a collection of time series pertaining to different views of the same object. For example, suppose we have five time series fragments for the same object for five different angles of rotation around the Z axis. The RNN can then be trained at each time t on all five data *streams* simultaneously, with the i-th copy of the RNN corresponding to the specific angle of rotation a_i. The same procedure of Section 2.1 is employed with the corresponding increase of the parameter S.

3. Results

We carried out numerous experiments to evaluate this method. The results described here are obtained when training the RNN on 225 objects and testing on an independent set of 96 objects. The RNN has 15 recurrent nodes in its first hidden layer, 8 recurrent nodes in its second hidden layer, and one non-recurrent output node. The training data set has approximately 20 million data points, and the average time series (representing one 3D object) has approximately 100,000 data points (time steps). The vehicle class has 85 examples, and each class has approximately the same number of data points. The majority of vehicle examples is of mid- and full-size sedans, but there are several SUVs and even buses as illustrated in Figures 7 and 8). The non-vehicle class is represented by trees, bushes, gates, parts of building and other obstacles as illustrated in Figure 9.

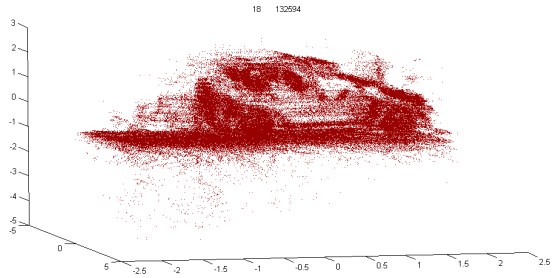

Figure 7. Example of an SUV as "seen" by the 3D lidar (normalized view).

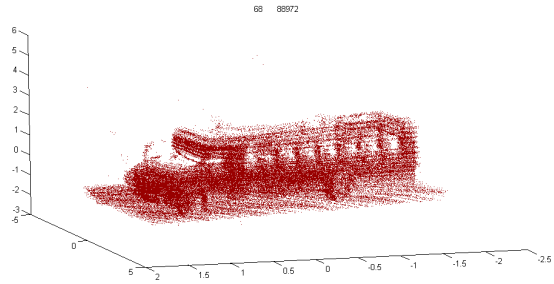

Figure 8. Example of a bus as "seen" by the 3D lidar (normalized view).

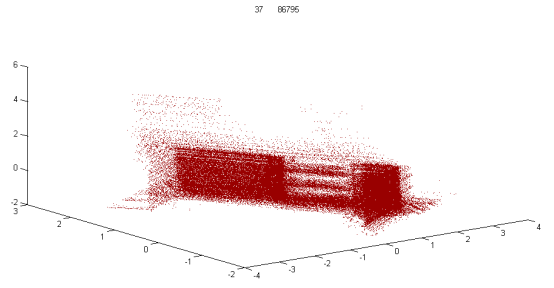

Figure 9. Example of a structure as "seen" by the 3D lidar (normalized view).

For examples of the RNN output please see Figures 4

and 5 of the previous section (top panels). A typical test confusion matrix is shown in Table 1.

	vehicle	non-vehicle
true vehicle	96.1%	3.9%
true non-vehicle	3.0%	97.0%

Table 1. Test confusion matrix

Figure 10 shows the results of the RNN trained with mitigation of rotational sensitivity as described in Section 2.2. The results of the same RNN trained without rotational invariance (baseline RNN) are also shown for comparison. The rotation invariant RNN shows much less sensitivity over the entire range of the viewing angles that the baseline RNN, and its average error is 39% better than that of the baseline RNN. It is still remarkable that even the baseline network performance is only mildly sensitive to very significant changes of the viewing angle.

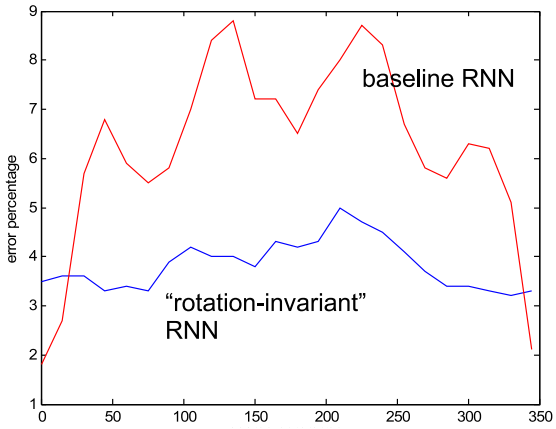

Figure 10. Performance comparison of two RNNs - baseline RNN (red; trained without explicitly encouraging rotational invariance) and the rotation invariant RNN (blue). The average error rates on the test set are 6.2% and 3.8%, respectively. The RMSE varies from 0.26 for lower rates to 0.55 for higher rates. The plot resolution is 15 degrees of rotation along the horizontal axis, but the plotted points are connected in a line for viewing convenience.

The training process is quite different from typical NN training process. In the NN training process typical of a weight update method based on the full batch of data (entire training set), the RMSE decreases steadily until it reaches a local minimum. We use a small batch procedure which relies only on a small subset of training data for each epoch, effectively exploiting the redundancy frequently present in large data sets and introducing stochasticity (due to random assignment of time series segments) which helps the training process to escape unsatisfactory local minima. That is why the RMSE behavior tends to be oscillatory, but the trend reflects its general decrease. The typical final RMSE is between 2.5 and 3 times smaller than the initial value of approximately 1.0. In addition, we monitor the RMSE and save the weights corresponding to the current lowest value of the RMSE for further testing. Figure 11 illustrates a typical training process for our RMLP.

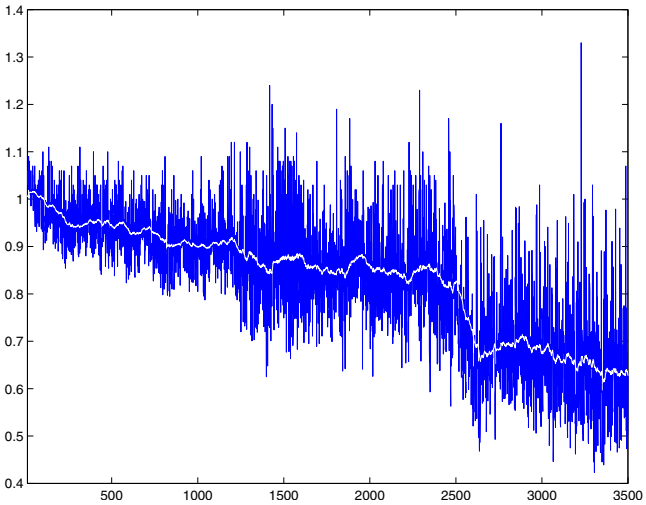

Figure 11. Typical training process of our RMLP (the trend is shown as white curve). The best weights are saved at epoch 3304 with the RMSE=0.423 with the error rate of 5.1% errors (approximately 2% vehicle misdetection and 3% false alarms).

4. Discussion

Low values of RMSE on the training set correspond generally to low percentage of errors not only on the training set but also on the test set, as revealed by preliminary experiments with data subsets. The expected behavior of the network output confirmed in experiments is to exhibit long periods (hundreds of time steps) of staying positive or negative and close to +1 or -1, respectively; see also the top panels in Figures 4 and 5 for the examples of the described behavior. Wildly oscillating network output usually signals the network's lack of confidence. This rarely happens to the well trained network on the training set, but can still occur during testing, e.g., because of inadequate segmentation. The variance of the network output could then serve as a confidence indicator of the network performance.

We wish to make several comments regarding RNN training and results:

• To avoid training conflicts, the data segmentation for RNN training should be done without objects of the different classes appearing within the same segmentation box. Naturally, at least some ground points will always be present even in the segmentation box occupied by a vehicle example, but from our past experience the RNN is known to be able to ignore minor label inconsistencies in data, as well as

to cope with small amounts of drop-outs and occlusions.

• For RMLP training, we ended up using the following parameters: $H_p = 3000$, $T = 3200$, $h = 100$, $S = 100$, and 3500 as the total number of training epochs. We carried out limited experimentation for training parameter optimization. We found that $H_p = 500$ and $T = 700$ are still adequate for successful RMLP training, whereas $H_p = 200$ and $T = 400$ are not. The truncation depth $h = 100$ seems to be larger than needed because the results for $h = 20$ are only slightly worse than those for the longer h. In addition, $S = 20$ seems insufficient for successful training. Much longer lengths, e.g., $H_p = 5000$ and $T = 5200$ or $T - H_p > 200$, only result in significantly longer training. (Likewise, much larger RNN, e.g., 25 recurrent nodes in the first hidden layer, and 10 recurrent nodes in the second hidden layer, did not yield performance improvements sufficient to justify substantial increase of training and execution complexity.) Thus, the RMLP appears to use only relatively short-term memory to recognize objects. This observation is also confirmed by testing the network when starting not in the very beginning of the time series but at arbitrary entry points spread over the entire length of the time series. Test results are affected little by changes of the starting point, by less than 1% in the error rate. Such insensitivity is understandable if we recall how the RNN training process is done (by training on randomly chosen segments within the time series) and what it is trying to achieve (to create "signature" classifier). (The initial values of the recurrent nodes of the RMLP are set to zeros at every entry point (see Step 2 of the training procedure in Section 2.1), preventing the network from using the data from the beginning of the time series to the current entry point.)

• It is not clear whether the RNN really needs long-term memory (other than the memory represented by its weights) to be successful for this task. The data for non-vehicle class is much richer in terms of the shapes and forms of objects than the vehicle class data. Besides, it is not useful to do a very fine segmentation of the vehicle class data to single out the vehicle object accurately because such segmentation would imply the use of knowledge unavailable during testing phase. For coarsely segmented vehicles (as is our case illustrated in Figures 2, 7 and 8) the data label may well be wrong in parts of the time series representing other objects adjacent to the vehicle, e.g., ground. If the network retains strong memory of its distant past and makes mistakes early on in the time series, such mistakes may force the network to continue making mistakes throughout the time series – the situation which our RMLP avoids.

• An RNN consisting of feedforward part with output-to-input feedback instead of RMLP with its internal feedback is not meaningful for our classification task. Indeed, the correct output is supposed to be quasi-constant throughout the time series pertaining to the given segmentation box. Such an output carries little information if fed into the network as another input.

• The advantage of our RMLP without explicit delay line at the input is that it can create input delays implicitly through the recurrent node feedback. Furthermore, computational complexity of multi-dimensional input delay line quickly becomes a serious implementation issue.

5. Conclusions and future work

We presented a new method to classify objects represented as collections of laser data points in 3D. The method works on normalized 3D object coordinates represented as temporally ordered inputs to a recurrent neural network which is trained by a second-order gradient based method. We also demonstrated an effective approach to mitigate sensitivity of the RNN performance to variations of the object viewing angle.

While at first glance the proposed method might look not practical from the standpoint of its robustness, the results of the rotation invariant training (Figure 10) do suggest that it is possible to make the method robust through a special training process. (The process results in an RNN robust to rotation around the vertical axis, which is the most critical rotation for objects on the relatively flat ground.) Furthermore, the proposed principle of recognition based on temporal "signature" of the object originally represented as a cloud of laser points is worthy of attention. The proposed method assumes only a coarse segmentation, i.e., an approximate selection of region of interest for which a classification decision is desired. In contrast, the spin image method requires not only more careful segmentation but also selection of key/feature points around which spin image projections are created – not-trivial task. In our method the system processes all the points in a relatively coarse segmentation box and infers the object's class automatically.

Future work will include: 1) comparison between different training methods, RNN architectures and the spin image method, and 2) integration of multi-sensor data for improved performance.

References

[1] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, Morgan Kaufmann, 3rd Edition, 2004.

[2] S. Frintrop, *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*, Morgan Kaufmann, 3rd Edition, 2004.

[3] Stephan Kirstein, Heiko Wersing, Edgar Körner, "A biologically motivated visual memory architecture for online learning of objects", *Neural Networks*, Vol. 21, 2008, pp. 65-77.

[4] SICK Sensor Intelligence, LMS-291 and similar sensor specifications are available from `http://www.sick.com`

[5] Andrew E. Johnson and Martial Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D scenes,"

IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, 1999, pp. 433–449.

[6] Hui Chen, Bir Bhanu, "Human Ear Recognition in 3D," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, no. 4, pp. 718–737, Apr. 2007.

[7] Toth, C. K., Barsi, A., Lovas, T., 2003. Vehicle recognition from LiDAR data. *International Archives of Photogrammetry and Remote Sensing,* Vol. XXXIV, Part 3/W13, pp. 162–166.

[8] Gregory Arnold, Timothy Klausutis and Kirk Sturtz, "3D laser radar recognition approaches," in *Computer vision beyond visible spectrum,* by Bir Bhanu and Ioannis Pavlidis (eds), Springer 2004, pp. 71–114.

[9] Philipp Lindner and Gerd Wanielik, "3D LIDAR Processing for Vehicle Safety and Environment Recognition", *Proc. SSCI 2009/Computational Intelligence in Vehicle and Vehicular Systems (CIVVS)*, Nashville, TN, March 30 – April 2, 2009.

[10] 3D lidar specifications are available from `http://www.velodyne.com/lidar`

[11] DARPA Urban Challenge 2007 homepage, `http://www.darpa.mil/grandchallenge/index.asp`

[12] `http://andrewnorton.info/2007/10/the-academic-we/`

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 2nd Edition, 1999.

[14] D. V. Prokhorov, G. V. Puskorius, and L. A. Feldkamp, "Dynamical Neural Networks for Control," see Chapter 16 in *A Field Guide to Dynamical Recurrent Networks*, J. Kolen and S. Kremer (eds), IEEE Press, 2001, pp. 257–289.