Application of Deep Learning to Route Odometry Estimation from LiDAR Data

Miguel Clavijo, Francisco Serradilla, José E. Naranjo, Felipe Jiménez and Alberto Díaz
University Institute for Automobile Research (INSIA)
Universidad Politécnica de Madrid (UPM)
Madrid, Spain
e-mail: miguel.clavijo@upm.es

***Abstract*— The Deep Learning techniques are a powerful tool to support the development of all sorts of information classification or processing techniques within the area of intelligent vehicles, since they are able to emulate the performance of the human brain when learning from experience. Specifically, the technique of Convolutional Neural Networks (CNN) has been successfully used in applications for classification and localization of pedestrians and obstacles on the road. However, CNN allow not only classification and pattern learning, but can be used for regression or modeling, like other kind of classical neural networks. The fundamental difference of both applications is that, while in classification the values of the network output are usually discrete, in regression or modeling applications the network can generate a continuous output with real numbers, allowing it to emulate the output of any type of system that is presented in the training set, with all its associated advantages, such as generalization and correct characterization of situations that have not learned explicitly. This paper presents an application of CNN for modeling in Intelligent Vehicles field, whose objective is to calculate the navigation parameters of a vehicle from the information supplied by a 3D LiDAR mounted on a vehicle that circulates in urban areas. Specifically, the developed CNN is able to calculate the speed and heading of a vehicle circulating in real time from the distance data supplied by the LiDAR sensor. The results show that the network is able to learn to calculate the speed and the yaw rate from the identification of the characteristic points of the environment, providing data that can be used to support the navigation of the vehicles.**

Keywords - deep learning; autonomous vehicle; odometry; LiDAR.

I. INTRODUCTION

Intelligent vehicles are characterized by equipping a large number of sensors and computer and communication systems, capable of providing all the information required for some advanced driving assistance systems, such as cooperative systems, specifically on the field of autonomous vehicles navigation. This large number of sensors allows monitoring the driving environment with high precision, even beyond the visual horizon due to the communications systems and, at the same time, supporting a safe navigation. However, this instrumentation has two fundamental problems. On the one hand, the high number of sensors (Computer vision, 3D LiDAR, Ultrasounds, Radar, Gyro, Compass, GPS, etc.) requires a large investment in these vehicles, increasing production costs. On the other hand, many of these sensors provide redundant information, which may or may not be used by the system, which in many cases underuse this data.

In this way, one of the most widely used sensors in the environment recognition of intelligent vehicles is the Laser Scanner 3D or LiDAR. This type of sensors usually provide an array of points with the distances from the sensor to the different elements of the environment in a range of 360º. Generally, this accurate information of the driving environment is used for the detection of pedestrians, vehicles or other obstacles on the road. However, since this precise information about the environment is available, it is possible to use it for other applications, such as navigation support [1] or visual odometry.

Visual odometry has been one of the last research fields to take part into the autonomous navigation applications. In general terms, visual odometry techniques tackle the SLAM (Simultaneous Localization And Mapping) problem, estimating the vehicle ego-motion and locating it in an unknown environment by using perception sensors as main source of information. Both estimating an accurate motion and tracking the vehicle route are two of the most difficult tasks in robotic and therefore in autonomous vehicle development. Solving the SLAM problem allows to perform critical tasks, such as the autonomous navigation where GPS signal can be lost or driving through complex areas, among others. In recent years, different visual odometry techniques have been developed using computer vision, stereo vision [2], LiDAR [3] or a sensory fusion between them [4][5]. Each of these algorithms, extract specific features of the environment such as flat surfaces, vertical corners, sharp angles, etc., from the data supplied by the sensor in each case and followed by a matching process between frames. Furthermore, in some cases it is necessary to make use of external motion sensors (e.g., IMUs, GPS/INS) in order to decrease the error. Another solution employed is the “loop closure” method when the incremental errors over time produce some drift. This implies that it does not work in real-time.

On the other hand, one of the most promising techniques for application, in multiple domains in general and in intelligent vehicles in particular, is Deep Learning. Deep Learning comprises a set of intelligent and bioinspired techniques based on neural networks with multiple hidden layers (usually more than 3). Convolutional Networks, autocorrelators, deep belief networks and Long Short-Term Memory (LSTM) are the four basic neural network techniques that establish the Deep Learning framework. Convolutional layers in a network are specialized in image processing and are those that learn the convolutions to perform on the input data (image pixels) to filter it or obtain relevant features. Specifically, the convolution processes an input image to obtain its relevant features. Adding several Full Connected traditional layers to the convolutional layers, the resulting network is able to identify patterns in the images, classifying them as belonging or not to a particular class or standard (e.g., pedestrian or car), and are widely used as classifiers for camera-based systems in the field of intelligent vehicles.

Due to all these characteristics, CNN is starting to be used in autonomous vehicles applications. As mentioned, this type of networks is mainly used in classification and pattern learning when the number of outputs is discrete. Because of this, several works have implemented CNN based system to identify obstacles in autonomous navigation [6]. On the other hand, CNN are not only being used for classification, but also for estimating the vehicle ego-motion. In this way, according to [7], using stereo vision and extracting features, is possible to estimate vehicle velocity and direction, even though it calculates only discretized values. In turn, in [8] LiDAR information is used to estimate odometry using regression, which entails, continuous values as output. However, in that work, voxel grids are used to extract generic features.

In this paper, a novel application of convolutional neural networks within the field of intelligent vehicles is presented. Thus, a CNN based system is proposed that is able to calculate the speed and yaw rate of a vehicle that circulates in urban areas at speeds up to 50 km/h. Specifically, it is proposed to use this type of networks using as input the information of the point cloud provided by a 3D LiDARIn addition, using this type of Deep Learning technique, allows a better understanding of the potential that CNNs have. Furthermore, this special application for autonomous vehicles is crucial, due to the high computational cost of the system when using a great number of data and classic programming methods. In this way, the CNN used is able to supply a precise output of the navigation parameters of the ego-vehicle and reduce drastically the computational cost once the network model is trained. Pre-training or environment features extraction is not previously made. This system has been implemented, trained and tested in the facilities and with the vehicles of the University Institute of Automobile Research (INSIA) of the Technical University of Madrid, obtaining results comparable to the data supplied by high performance speedometer, GPS and gyroscope.

The paper is structured as follows, in Section II, the architecture of the CNN used is described as well as how the training process has been tackled. In Section III, the procedure for transforming the raw data to the image-data for the network input is adressed. The results of speed and yaw rate obteined in the test dataset are shown in Section IV. Finally, in Section V, the conclusion and further works are dicussed.

II. CONVOLUTIONAL NETWORK ARCHITECTURE, PARAMETERS AND TRAINING FRAMEWORK

A. *Architecture description*

A Convolutional Neural Network is a feed forward neural network mainly composed of convolutional layers. Those layers are the matrices shown in (1)

$$H \times W \times C \tag{1}$$

Where H and W are spatial dimensions and C the number of channels (or channel dimension).

The architecture presented here is the result of a number of trial-and-error steps with different network sizes (both number of layers and convolutional networks dimensions), learning methods and parameters. The final architecture is summarized in Figure 1. The network is made up of six layers – four convolutional layers and two fully connected.

The input to the network is presented as 8×300×2 images, where the two channels corresponds to the image created from the points data captured by the LiDAR at "*t*" and "*t-1*". Then, features are extracted through the convolutional layers, adjusting the resulting features to the final two values (speed and yaw rate) using two fully connected layers.

Each convolutional layer is composed of three or four operations or sub-layers:

- The convolution operation itself, where, given an image with the pattern defined in (1), a set of filters ($N \times M \times C$ matrices where N≤H and M≤W)

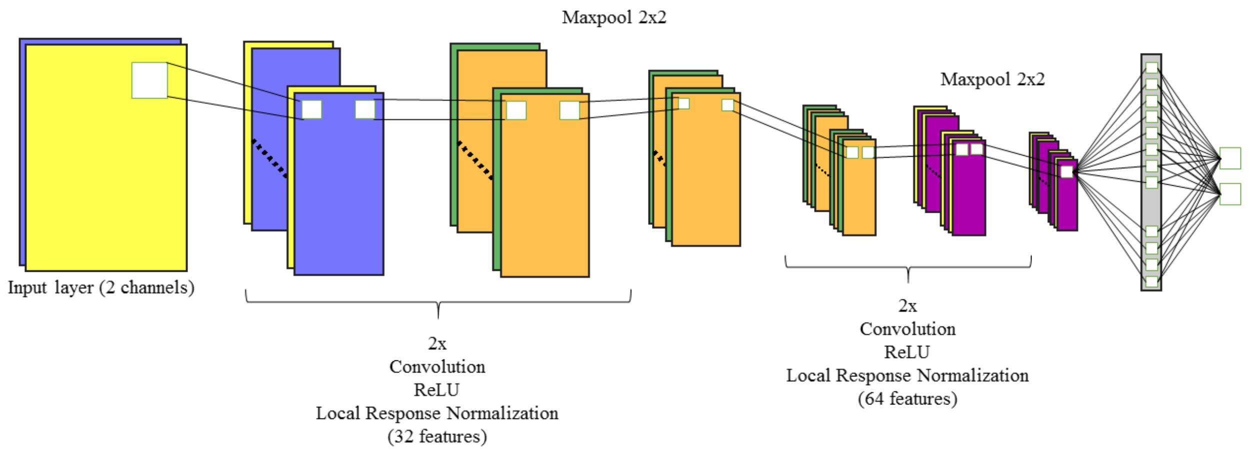

Figure 1. Convolutional network architecture proposed.

traverse the image with a step of 1 extracting 1 value for each sub-image.

- A Rectified Linear Units (ReLUs) [9] network, with neurons whose activation function is:

$$f(x) = \max(0, x) \tag{2}$$

Its purpose is to introduce non-linearity to the net. Other functions like (3) and (4) can also be used but they are much slower in terms of training time than ReLUs [9].

$$f(x) = \tanh(x) \tag{3}$$

$$f(x) = \frac{1}{1+e^{-x}} \tag{4}$$

- Local response normalization operation [10], whose aim is to increase the differences (i.e., improve the contrast) between adjacent pixels.

- Maxpool operation, is a non-linear subsampling method where a filter and a step are defined (usually a N×N filter with a step of N), replacing each subset of that window size by its maximum value.

The first and second convolution layers have a dimension of 3×15×32. The maxpool operator is applied at the end of the second layer with a 2×2 dimension with a step of 2, so the output is an image of dimensions 4×150×32, that is, 32 features. Third and fourth convolution layers have a dimension of 2×5×64. As with the second layer, a maxpool operator of same dimensions and step size is applied at the end of the fourth layer so the output for this layer has the dimensions 2×75×64, then 64 features.

The fully connected layers are traditional feed-forward neural networks where all the outputs of one layer are connected to all the inputs of the next one. In this case, the first fully connected layer transforms the 2×75×64 outputs of the last convolutional layer to a vector of 512 values. This outputs are then transformed into the last 2 values by second fully connected network.

Finally, a learning-with-dropout scheme has been used, with a dropout rate of 0.9 (10% of the neurons are removed each iteration). The dropout technique [11] is a way to avoid overfitting by randomly removing some of the neurons in the network each iteration. This way of learning makes the neurons not to learn by memory, but sharing knowledge among several instead.

B. *Learning operation*

The proposed model was trained using a stochastic gradient-based optimization algorithm called Adam [12] with a learning rate of $1e10^{-5}$ throughout 3,000 iterations. It has been tested with different learning rates: $1e10^{-3}$, $1e10^{-4}$ and $1e10^{-5}$. With a learning rate of $1e10^{-3}$ and $1e10^{-4}$, the convergence behavior obtained suggested that it was too high for the topology of the particular problem. Therefore, it has been chosen to use this value of learning rate.

The network was developed in the Python programming language with the help of the TensorFlow library [13] for the modelling and parallel training setups.

For the selected training dataset, this convolutional network configuration took 55 hours on a computer with GNU/Linux, a Xeon E3-1200 (family) v3/4th Gen Core Processor and a NVIDIA GTX 980Ti with 6GB and 2816 CUDA cores. As mentioned above, the execution time once the network is trained, allows the performance in real time.

III. DATASET AND TESTS

The architecture of the described network uses raw data obtained by the 3D LiDAR as input. This makes possible that the implementation of the trained network in the on-board computer be simpler and not dependent on extra sensors. As outputs, this network estimates the vehicle speed and its yaw rate.

In order to create the dataset of the network, a pre-treatment of the LiDAR data was performed. As ground truth, it has been used the speed data available from a speedometer and the yaw angle acquired by a high accurate Gyro, placed on the vehicle.

For a neural network training, it is necessary that the number of inputs and outputs is the same for all the dataset. Furthermore, when using convolution networks, a type of images has been developed from where the features are extracted.

The creation of these data-images requires several processing steps of the raw data of the laser scanner for each frame:

- First, the laser field of view (FOV) range is defined. Vertically, the range of vision is set between 3º and -11º, dividing the laser points in 8 rows. The horizontal FOV comprises two 150° areas located on both sides of the vehicle and placed centrally on its transverse axis (Figure 2). This range of vision was chosen given that it contains the area in which the environment perception generates the better amount of information for the estimation of odometry.

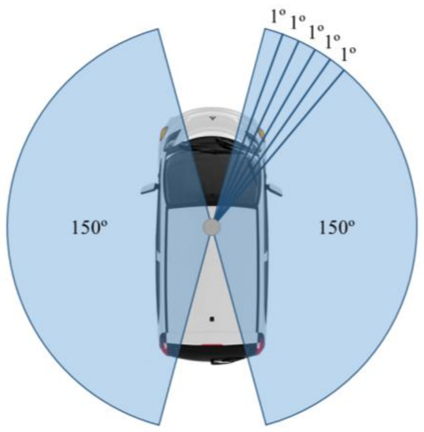

Figure 2. Horizontal FOV and its discretization.

- The LiDAR sensor used does not guarantee the generation of the same number of point in each revolution. Thus, in order to obtain the same amount of the data in each frame, the horizontal FOV is discretized with an amplitude of 1º (Figure 2). The distance of the LiDAR-points is acquired and these points are classified according to its horizontal angle and the channel through they were obtained. For this reason, 300 values are acquired in each of the 8 channel defined previously. This will be the data image resolution. In the case that more than one point exists for the same "pixel", the one presenting the lowest distance value remains.

- The disadvantage of LiDAR versus computer vision technology concerns the data dispersion when the distance is greater. Due to this, obtaining data that does not satisfy the conditions of the "pixel" is a possibility. Therefore, when a "pixel" with a null value is acquired in the image, a linear interpolation between the near valid values is done. In addition, as mentioned in other works [14], points density per length unit is inversely proportional to the distance to the sensor. Thus, in order to compensate the values distribution, a logarithmic filtering is applied in each pixel-value, followed by a normalization between 0 and 1, according to the distance. As a result, the histogram after this calculation is widely distributed throughout the normalized distance range (Figure 3).

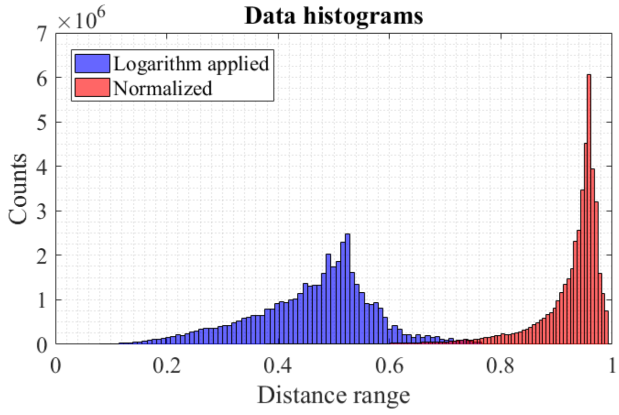

Figure 3. Data with and without logarithm applied.

As a result, in Figure 3 the original normalized data is shown in red. This value distribution is concentrated in a narrow range (between 0.8 and 1). In contrast, the data distribution after apply the logarithm to widen along the entire range is represented in blue.

- Finally, two data images are created, one for each channel of the network (parameter C in (1)) (Figure 4). The first corresponds to time "*t*" and the second is the image defined for time "*t-1*". The data acquisition is at 10 Hz, so the time transition is 0.1 s.

  As outputs, the average speed and the yaw rate during the transition time are included.

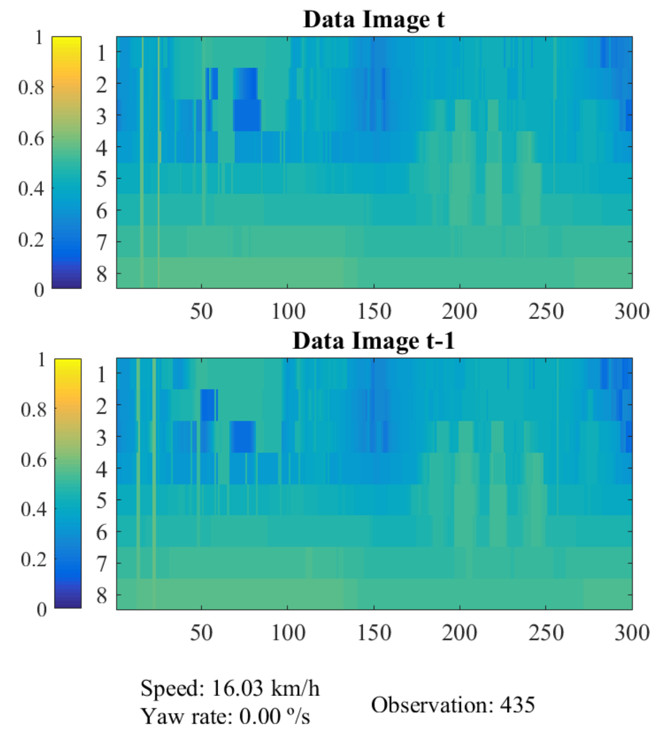

Figure 4. Example of data-images in a specific observation.

Figure 4 represents one of the images create for the CNN. The horizontal range of vision discretization results in 300 divisions, while the data is adquised by 8 channels corresponding to the vertical range of vision, defined between -11º and 3º. Therefore, the image size is 8×300 pixels. Each of those pixels is the distance data of the LiDAR points after the logarithm has been applied and normalized.

Neural network training requires a large amount of observations. This is why create a dataset for Deep Learning applications take a considerable amount of time. As a solution for this problem, a "mirror function" was implemented. This function doubles the number of observations used for training, obtaining twice the number of valid samples and taking into consideration new situations in the environment. This is possible because the "mirror function" calculates the inverse of data-images and their outputs, simulating a data acquision performed in a non-real scenario whose trajectory is the inverse of the real-scenario.

IV. RESULTS AND DISCUSSION

Several data collections were performed and the correct behavior for the network training was proven.

In this case, a route at Campus Sur - UPM has been selected, which corresponds to a one-way street with 2 lanes and cars parked on both sides of the road (Figure 5).

Figure 5. Urban area selected for the CNN training set.

Specifically, a data collection for the training set during one hour was made, the equivalent to 25.6 km in an urban environment. During this data collection, the urban scenario had dynamic traffic, tight bends, diaphanous areas, buildings, etc. In addition, the driving mode was different throughout the entire test, with varying speeds, changing lanes and making stops. The training data collection consists of 34,530 observations, which gives rise to a total of 69,060 samples when applying the “mirror function” mentioned above.

Regarding the dataset test for the network, a different route was chosen, where one part corresponded to the same zone as the training set and the other part was an unknow area. This dataset test comprises 2.85 km or 4,756 observations.

Once the network training was finished, the CNN model was applied to the dataset test. The results obtained are depicted in Figure 6 and Figure 7 for the vehicle speed and its yaw rate, respectively.

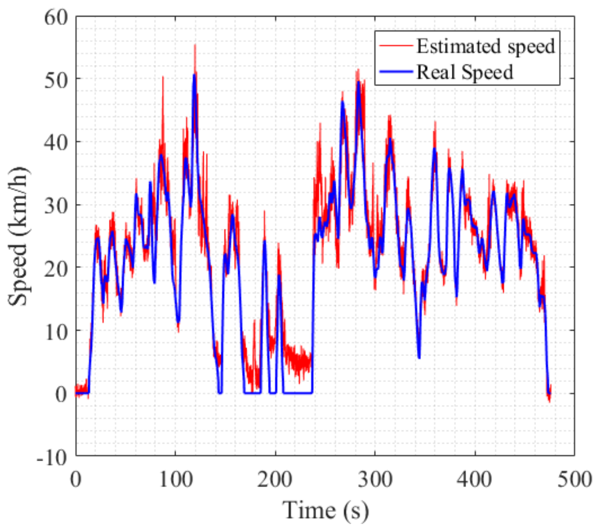

Figure 6. Speed estimation in test dataset.

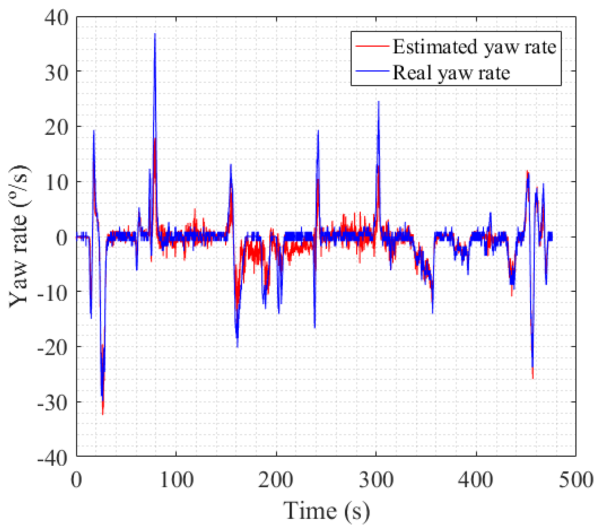

Figure 7. Yaw rate estimation in test dataset.

It can be considered that the estimation of the speed is accurate, mainly focusing on the accelerations and decelerations, where the value estimated by the network and the real value have the same outline, obtaining a Root Mean Square (RMS) error after 3,000 epoch of 3.61 km/h.

Concerning the yaw rate, the estimation is quite precise, being better adjusted to the actual data when there are concatenated path changes. An RMS of 2.8 °/s is obtained of the whole test after 3,000 epoch.

Both Figure 6 and Figure 7, illustrate that between the time instants 150 s and 250 s the estimation of the values and the real data differs somewhat more. This is largely due to the fact that it corresponds to the unknown area in which it had not previously circulated and which did not correspond of the training dataset, as previously described. On the other hand, the network was able to generalize unknown surroundings and the performance in this specific situation is reasonably successful.

It must be mentioned that the RMS obtained are in the dataset test and that it is different from the learning RMS, which has been 0.92 km/h and 0.94 °/s for the whole training set.

It can be assumed that the network, with a short training set, is capable of learning and generalizing. However, the CNN should be taught with training sets that include other road types to be able to operate in all cases. Not a great number of areas, but situations that serve as models.

V. CONCLUSION

In this paper, an application of Deep Learning techniques to estimate the visual odometry of an autonomous vehicle has been presented. Specifically, the information provided by a 3D LiDAR has been used as input of a Convolutional Neural Network with a novel architecture that is able to estimate the values of speed and yaw angle. This system has been implemented, trained and tested in the facilities of the

University Institute for Automobile Research, using its Campus as testbed area. The results of the implementation and commissioning of this system is that the CNN is able to successfully estimate the output values, generalizing correctly when in situations that have not been previously learnt by the network. Two conclusions have been achieved of the work presented in this paper: on one hand, a novel application of CNN for regression in visual odometry has been developed. On the other hand, the results of this system show that the network need typical road areas to learn how to estimate the navigation parameters and, from these data, is able to generalize the navigation of different areas not learnt.

In future works, the main aim will be improving the output precision, collecting data of specific environments and traffic situations. Also, changes in the network architecture here described, are providing useful information about the contribution of each layer. For that reason, adding additional convolutional layers may improve the overall performance.

ACKNOWLEDGMENT

This work has received the support of the European Project AUTOCITS (INEA/CEF/TRAN/M2015/1143746. Action No: 2015-EU-TM-0243-S), MINECO CAV (TRA2016-78886-C3-3-R), and Excellence Network SEGVAUTO-TRIES. The authors would also like to especially thank Óscar Gómez, who as member of the team, was responsible for setting up the test equipment and helped carrying out necessary tests to the development of this work.

REFERENCES

[1] M. Clavijo, F. Jiménez, J. E. Naranjo, and Ó. Gómez, "Supervision system for reversible lanes based on autonomous vehicles," in *11nd European Congress and Exhibition on Intelligent Transport Systems and Services*, 2016, no. June, pp. 6–9.

[2] I. Cvisic and I. Petrovic, "Stereo odometry based on careful feature selection and tracking," in *2015 European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–6.

[3] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Auton. Robots*, no. October 2014, 2016.

[4] F. Mutz, L. P. Veronese, T. Oliveira-Santos, E. de Aguiar, F. A. Auat Cheein, and A. Ferreira De Souza, "Large-scale mapping in complex field scenarios using an autonomous car," *Expert Syst. Appl.*, vol. 46, pp. 439–462, 2016.

[5] A. Tamjidi and C. Ye, "6-DOF Pose Estimation of an Autonomous Car by Visual Feature Correspondence and Tracking," *Int. J. Intell. Control Syst.*, vol. 17, no. 3, pp. 94–101, 2012.

[6] P. Christiansen, L. N. Nielsen, K. A. Steen, R. N. Jorgensen, and H. Karstoft, "DeepAnomaly: Combining background subtraction and deep learning for detecting obstacles and anomalies in an agricultural field," *Sensors*, vol. 16, no. 11, p. 1904, Nov. 2016.

[7] K. Konda and R. Memisevic, "Learning Visual Odometry with a Convolutional Network," in *International Conference on Computer Vision Theory and Applications*, 2015, pp. 486–490.

[8] A. Nicolai, R. Skeele, C. Eriksen, and G. A. Hollinger, "Deep Learning for Laser Based Odometry Estimation," in *Science and Systems Conf. Workshop on Limits and Potentials of Deep Learning in Robotics*, 2016.

[9] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, no. 3, pp. 807–814.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[12] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2014, pp. 1–13.

[13] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, 2016, pp. 265–284.

[14] F. Jiménez, M. Clavijo, J. E. Naranjo, and Ó. Gómez, "Improving the Lane Reference Detection for Autonomous Road Vehicle Control," *J. Sensors*, vol. 2016, p. 13, 2016.