JAISCR, 2017, Vol. 7, No. 4, pp. 243 – 255
10.1515/jaiscr-2017-0017

A MACHINE LEARNING APPROACH FOR THE SEGMENTATION OF DRIVING MANEUVERS AND ITS APPLICATION IN AUTONOMOUS PARKING

Gennaro Notomista[1], Michael Botsch[2]

[1] *Università degli Studi di Napoli "Federerico II", Via Claudio 21, 80125 Napoli, Italy*

[2] *Technische Hochschule Ingolstadt, Esplanade 10, 85049 Ingolstadt, Germany*

submitted: 14th April 2016; accepted: 14th November 2016

Abstract

A classification system for the segmentation of driving maneuvers and its validation in autonomous parking using a small-scale vehicle are presented in this work. The classifiers are designed to detect points that are crucial for the path-planning task, thus enabling the implementation of efficient autonomous parking maneuvers. The training data set is generated by simulations using appropriate vehicle–dynamics models and the resulting classifiers are validated with the small-scale autonomous vehicle. To achieve both a high classification performance and a classification system that can be implemented on a microcontroller with limited computational resources, a two-stage design process is applied. In a first step an ensemble classifier, the Random Forest (RF) algorithm, is constructed and based on the RF-kernel a General Radial Basis Function (GRBF) classifier is generated. The GRBF-classifier is integrated into the small-scale autonomous vehicle leading to excellent performance in parallel-, cross- and oblique-parking maneuvers. The work shows that segmentation using classifies and open-loop control are an efficient approach in autonomous driving for the implementation of driving maneuvers.

Keywords: autonomous parking, ensemble learning, maneuver segmentation

1 Introduction

Autonomous Mobile Robots have become recently a topic of great interest among more and more researchers in the world. Due to the huge developments done in the field of autonomous navigation and motion planning, the perspective of full autonomous robots has already become real.

By now, autonomous mobile robots have been used in many fields and services, first among everything is industry. But they are used, for instance, in underwater exploration too, aerial surveys are accomplished with the so-called *Unmanned Aerials Vehicles* and space robots recently managed to land on a comet. Robotics is successfully applied more and more also to agricultural systems, to construction and to medicine.

Due to legal issues, however, the knowledge aquired in these fields cannot be fully transfered to *autonomous driving*. According to the UN/ECE Regulation R 79 for steering systems automatic steering is only allowed at speeds below 10 km/h. It is worth to notice that today, most of the cars are equipped with all the sensors which would be necessary to make them completely autonomous. In fact, on a modern car there are already many exteroceptive sensors (such as radar, lidar, camera, infrared and ultrasound sensors), besides the inertial measurement unit and the wheel speed sensor that have been playing a big role in the estimation of the car movement already for several years (in ABS and ESP, for example). Until the legislation will change a first application of a full-autonomous-driving vehicle can be found in *autonomous parking*.

A lot of work has been already done in this direction, making the autonomous parking already a feature in production cars. Therefore, many methods to evaluate the best parking maneuver have been presented. In [1], for instance, the parking maneuver, both for parallel and for cross parking, is divided into several segments, which are calculated analitically according to the geometry of the parking area. In [2] the shape of the maneuver is adjusted online according to the range data coming from exteroceptive sensors. In a more general motion planning scenario, see [3], a *Linear Quadratic Regulator* (LQR) algorithm can be used to minimize a cost function in order to find the most suitable inputs needed to follow the maneuvers. Other solutions can be founded in [4], [5], [6], [7] and [8], where *Artificial Neural Networks* (ANN), *Radial Basis Functions Networks* (RBFN) and *Rapidly-Exploring Random Trees* (RRT) are used to generate and learn both acceleration and steering commands. Finally, in [9] and [10], other two different techniques are used to solve the same problem, i. e., hybrid fuzzy controllers and point-stabilization of non-holonomic systems.

The approach presented here is different since pre-defined sub-maneuvers of a parking maneuver are used for parallel and for cross parking. A sub-maneuver consists of a segment that is characterized by specific values of the steering angle (as explained in detail in the Subsections 3.2.1 and 3.2.2). The segmentation points are those that are learned by a machine-learning algorithm. Then, based on measurements, coming from the exteroceptive sensors, and on the position of the vehicle in a global reference frame, estimated by using a extended Kalman filter, the relative location of the vehicle with respect to the parking spot is evaluated and a machine learning algorithm is triggered in order to find out whether the current position is a good segmentation point or not.

This paper is organized as follows. In Section 2 there is a detailed explanation of the algorithm that runs on the vehicle throughout the parking maneuver and an overview of the technical background that is necessary for the algorithm. In Section 3 the machine learning technique applied to the considered scenario is described. In Section 4 the results of the presented algorithm are illustrated. For this purpose a small-scale autonomous vehicle has been used. After a brief explanation of its hardware and software architecture, the results of the implementation of the parking procedure are shown.

Throughout this work, vectors and matrices are denoted by lower and upper case bold letters, and random variables are written using sans serif fonts.

2 Architecture of the Proposed Parking Scheme

Figures 1 and 2 show the hierachical architecture of the algorithm that runs during the parking maneuver.

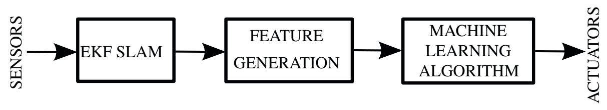


Figure 1. Algorithm architecture

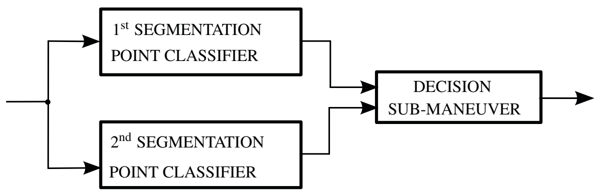

1st SEGMENTATION POINT CLASSIFIER
2nd SEGMENTATION POINT CLASSIFIER
DECISION SUB-MANEUVER

Figure 2. Block “MACHINE LEARNING ALGORITHM” of Figure 1

A global reference frame is defined and with the aid of an Extended Kalman Filter (EKF), the position of the moving vehicle and of its environment are estimated. At the same time, a suitable parking spot is sought. As soon as this is found the parking maneuver itself begins. The car starts moving according to the first segment of the maneuver and its position relative to the parking spot is forwarded to the the “FEATURE GENERATION” block and then to the “MACHINE LEARNING ALGORITHM” block in order to determine whether the next sub-maneuver should start. For the maneuvers considered in this work the machine learning block consists of two classifiers which divide a parking maneuver in three sub-maneuvers. The block "DECISION SUB-MANEUVER" is assuring that the sub-maneuvers are implemented in the desired chronological order. As soon as a transition to a new sub-maneuver is detected by the classifiers the actuators are controlled to realize the new sub-maneuver.

2.1 Vehicle Dynamics Model

For the estimation of the vehicle position a statistical filter is used. This filter is based on the model of the system, i. e. the vehicle, whose state has to be estimated. In [11] there is a comparison of the models that can be used for vehicle tracking. In this work three of the most common models have been considered, that are the *two track model*, the *Continuous Turn Rate and Acceleration* (CTRA) and the *kinematic model*.

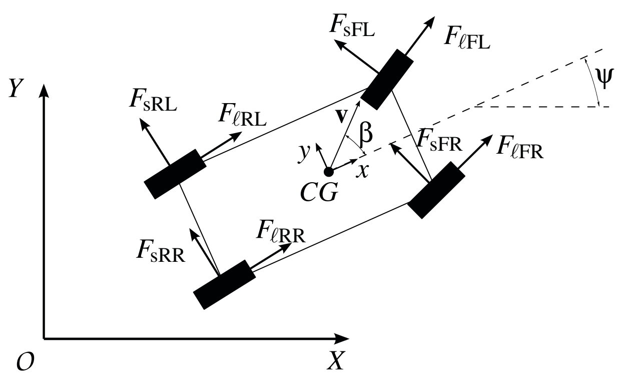


Figure 3. Parameters of the two track model

The first model, the most sophisticated one, is represented by the following differential equations describing how the vehicle state changes over time:

$$\begin{bmatrix}\dot{v}\\ \dot{\beta}\\ \ddot{\psi}\end{bmatrix} = \boldsymbol{f}\Big(m, I_z, \ell_{\mathrm{f}}, \ell_{\mathrm{r}}, w, v, \beta, \delta_{\mathrm{FL}}, \delta_{\mathrm{FR}}, \delta_{\mathrm{RL}}, \delta_{\mathrm{RR}} \\ F_{\ell\mathrm{FL}}, F_{\ell\mathrm{FR}}, F_{\ell\mathrm{RL}}, F_{\ell\mathrm{RR}} \\ F_{\mathrm{sFL}}, F_{\mathrm{sFR}}, F_{\mathrm{sRL}}, F_{\mathrm{sRR}}\Big), \tag{1}$$

where m is the vehicle mass, I_z its inertia around the z axis, ℓ_{f} and ℓ_{r} are the distances between the vehicle center of gravity and the front and rear axles, respectively, w is the width of the vehicle. δ_{FL}, δ_{FR}, δ_{RL}, δ_{RR} are the tire slip angles and $F_{\ell\mathrm{FL}}$, F_{sFL}, $F_{\ell\mathrm{FR}}$, F_{sFR}, $F_{\ell\mathrm{RL}}$, F_{sRL}, $F_{\ell\mathrm{RR}}$, F_{sRR} the longitudinal and side forces exchanged by the four tires with the ground. The quantities $\dot{v}$, $\dot{\beta}$ and $\ddot{\psi}$ are the velocity derivative, the slip angle derivative and the yaw acceleration, respectively, and they represent the vehicle state (see Figure 3). The advantage of this model is the high precision in the prediction of the vehicle behavior. This, on the other hand, requires the knowledge of several parameters to completely describe the system. Some of these parameters can be either unknown or difficult to estimate. An example is the friction coefficients of the tire/road contact needed to evaluate the tire forces using, for instance, the Magic Tire Formula

$$\begin{aligned} F_{\ell ij} &= F_{\mathrm{z}ij}\mu_{\ell ij}\sin\left(c_{\ell ij}\arctan(b_{\ell ij}\frac{s_{\ell ij}}{\mu_{\ell ij}})\right), \\ F_{\mathrm{s}ij} &= F_{\mathrm{z}ij}\mu_{\mathrm{s}ij}\sin\left(c_{\mathrm{s}ij}\arctan(b_{\mathrm{s}ij}\frac{s_{\mathrm{s}ij}}{\mu_{\mathrm{s}ij}})\right). \end{aligned} \tag{2}$$

A first simplification of this model can be achieved considering a constant acceleration, a constant turn rate and a small discretization time-step T. This leads to the CTRA model which is expressed by the following difference equations

$$\begin{cases} \mathsf{r}_X[n+1] = \mathsf{r}_X[n] + \mathsf{v}[n]\cos(\psi[n])T \\ \qquad - \mathsf{v}[n]\sin(\psi[n])\dot{\psi}\frac{T^2}{2} + \mathsf{a}_{\parallel}\cos(\psi[n])\frac{T^2}{2} \\ \mathsf{r}_Y[n+1] = \mathsf{r}_Y[n] + \mathsf{v}[n]\sin(\psi[n])T \\ \qquad + \mathsf{v}[n]\cos(\psi[n])\dot{\psi}\frac{T^2}{2} + \mathsf{a}_{\parallel}\sin(\psi[n])\frac{T^2}{2}, \end{cases} \tag{3}$$

where $\mathsf{r}_X[n]$, $\mathsf{r}_Y[n]$ denote the position of the vehicle's center of gravity at time step n in the global frame (X,Y), $\mathsf{v}[n]$ the velocity of the vehicle, $\psi[n]$ its yaw angle, $\dot{\psi}$ its yaw rate, and $\mathsf{a}_{\parallel}$ the longitudinal component of the vehicle's acceleration. Figure 4 visualizes this planar model of the vehicle. In the Figure δ denotes the wheel angle, i. e., the angle between the front wheels and the longitudinal axis of the vehicle.

To be noticed is also that, in both formulas of Eq. (3), the term $\mathsf{v}[n]\dot{\psi}$ is equal to $\mathsf{a}_{\perp}[n]$, the centripetal acceleration.

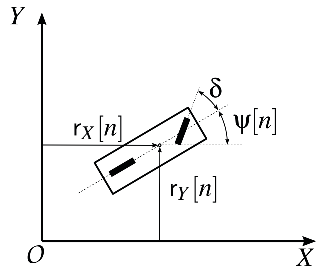


Figure 4. Vehicle parameters for the kinematic model

The formulas in Eq. (3) can be further simplified when the discretization time-step T is very small, such that $T^2 \approx 0$. Moreover the yaw-rate $\dot{\psi}$ can be expressed using the basic kinematic formula $\dot{\psi} = \mathsf{v}/R$, with R being the radius of the vehicle‘s trajectory. The radius R can also be approximated using a simple geometric model of the vehicle by $R = \ell/\tan(\delta)$, where ℓ is the wheel base and δ is the steering angle. This leads to the following set of equations

$$\begin{cases} \mathsf{r}_X[n+1] & = \mathsf{r}_X[n] + \mathsf{v}[n]\cos(\psi[n])T \\ \mathsf{r}_Y[n+1] & = \mathsf{r}_Y[n] + \mathsf{v}[n]\sin(\psi[n])T \\ \mathsf{v}[n+1] & = \mathsf{v}[n] + \mathsf{a}_{\|}T \\ \psi[n+1] & = \psi[n] + \mathsf{v}[n]\frac{\tan(\delta)}{\ell}T. \end{cases} \tag{4}$$

2.2 Extended Kalman Filter

The statistical filter that is going to be used is an Extended Kalman Filter.

The state vector at time step n is defined as

$$\begin{aligned} \mathbf{x}[n] &= [\mathsf{x}_1[n], \mathsf{x}_2[n], \mathsf{x}_3[n], \mathsf{x}_4[n]]^{\mathrm{T}} \\ &= [\mathsf{r}_X[n], \mathsf{r}_Y[n], \mathsf{v}[n], \psi[n]]^{\mathrm{T}}, \end{aligned} \tag{5}$$

where $\mathsf{r}_X[n]$ and $\mathsf{r}_Y[n]$ are the coordinates of the vehicle position vector expressed in the reference system depicted in Figure 4, $\mathsf{v}[n]$ is the longitudinal velocity of the vehicle and $\psi[n]$ is its yaw angle computed with respect to the X axis. All the quantities are referred to the time step n.

The input vector at time step n as

$$\boldsymbol{u}[n] = [u_1[n], u_2[n]]^{\mathrm{T}} = \left[a_{\|}, \delta\right]^{\mathrm{T}}, \tag{6}$$

where $a_{\|}$ has been introduced in the previous Section. The observation vector, at time step n, is

$$\mathbf{y}[n] = [\mathsf{y}_1[n], \mathsf{y}_2[n]]^{\mathrm{T}} = [\mathsf{v}[n], \psi[n]]^{\mathrm{T}}, \tag{7}$$

and the system noise is denoted by $\boldsymbol{\eta}_{\mathrm{s}}$, and the measurement noise by $\boldsymbol{\eta}_{\mathrm{m}}$.
In this way, we can write the state and the measurement equation in vectorial form as follows

$$\begin{cases} \mathbf{x}[n+1] & = \boldsymbol{f}(\mathbf{x}[n], \boldsymbol{u}[n]) + \boldsymbol{\eta}_{\mathrm{s}} \\ \mathbf{y}[n] & = \boldsymbol{h}(\mathbf{x}[n], \boldsymbol{u}[n]) + \boldsymbol{\eta}_{\mathrm{m}}. \end{cases} \tag{8}$$

As described in [12] or [13], in order to perform the extended kalman filtering, the Jacobians $\nabla \boldsymbol{f}$ and $\nabla \boldsymbol{h}$ are required, where $\boldsymbol{f}$ and $\boldsymbol{h}$ are the functions described in Eq. (8).
For the presented case where $\boldsymbol{f}$ is given by Eq. (4), the Jacobian is time dependent and at time step n it is equal to

$$\nabla \boldsymbol{f}[n] = \begin{bmatrix} 1 & 0 & \cos(\hat{\mathsf{x}}_4[n])T & -\hat{\mathsf{x}}_3[n]\sin(\hat{\mathsf{x}}_4[n])T \\ 0 & 1 & \sin(\hat{\mathsf{x}}_4[n])T & \hat{\mathsf{x}}_3[n]\cos(\hat{\mathsf{x}}_4[n])T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{\tan(u_2)}{\ell}T & 1 \end{bmatrix}. \tag{9}$$

The hatted quantities represent the components of $\hat{\mathbf{x}}[n]$, the estimate of the state vector $\mathbf{x}[n]$.
The output equation, on the contrary, is linear, $\nabla \boldsymbol{h}$ is time invariant and it is equal to

$$\nabla \boldsymbol{h} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

2.3 Localization

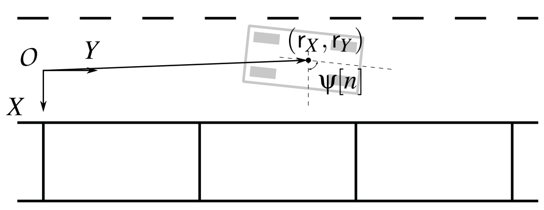


Figure 5. Global frame used for localization and maneuver planning

Figure 5 shows the global frame that is defined and used for localization and maneuver planning. It corresponds to that represented in Figure 4 and it is initialized at the beginning of the parking area in which the parking maneuver will be performed. For the case presented here, the EKF is used to incrementally estimate the vehicle state. These estimates are used as inputs to generate features (see Subsection 3.3) that represent the inputs to the machine learning algorithms, which decide whether the current vehicle state with respect to the map of the parking area is a good segmentation point for the parking maneuver or not.

2.4 Feature Generation and Classification

The segmentation classifiers in Figure 2 are implemented by machine-learning algorithms. Many relations that are found by statistical learning methods in data can be represented in the form of classification or regression functions. Classification and regression aim at estimating values of an attribute of a system based on previously measured attributes of this system. Given a set of measured observation attributes $\boldsymbol{p} = [p_1, \ldots, p_{N'}]^{\mathrm{T}} \in \mathbb{R}^{N'}$, statistical learning methods estimate the values of a different attribute z. If z takes on continuous numerical values, i. e., $z \in \mathbb{R}$ one talks about regression and if it takes on discrete values from a set of Q categorical values, called classes, i. e., $z \in \{c_1, \ldots, c_Q\}$ one talks about classification. Often a preprocessing of the observation vector $\boldsymbol{p}$ is performed in order to simplify the mapping from $\boldsymbol{p}$ to z. Preprocessing plays a very important role being a possibility to introduce *a priori* knowledge about the considered machine learning problem. This preprocessing transforms the observation vector $\boldsymbol{p}$ into the so-called feature vector $\boldsymbol{w} \in \mathbb{R}^N$. Defining feature vectors is the most common and convenient means of data representation for classification and regression problems. A pair $(\boldsymbol{w}, z)$ is called a pattern, $\boldsymbol{w}$ the “input” and z the “output” or “target”. Because the measured attribute values are subject to variations which often cannot be described deterministically, a statistical framework must be adopted. In this framework, $\boldsymbol{w}$ is the realization of the random variable w and z of the random variable z. One can think of the mapping from p to z or the mapping from w to z as a black box representing the process of interest.

In machine learning one is interested both in generating from the observation p a feature vector w that is suitable for the application at hand and in estimating the mapping from w to z. In supervised learning a set of M already known patterns, the so-called training set $\mathcal{D}$ is used

$$\mathcal{D} = \{(\boldsymbol{w}_1, z_1), \ldots, (\boldsymbol{w}_M, z_M)\}. \tag{11}$$

It should be noted that for a good performance of the learning system, which enables to predict accurately the output z for a new unseen measurement vector p, the construction of the feature vector w is extremely important. Figure 6 shows that the computed output $\hat{\mathsf{z}}$ can only be a good estimate of the target z corresponding to w if both the feature extraction and the mapping g from w to $\hat{\mathsf{z}}$ are chosen properly.

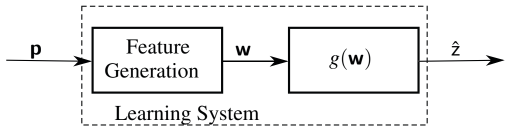


Figure 6. Learning system

The common approach in the statistical learning framework to find the mapping for classification

$$g : \mathbb{R}^N \to \{c_1, \ldots, c_K\}, \mathsf{w} \mapsto \mathsf{z} \tag{12}$$

is to minimize the risk that is defined as the expectation of a loss function $\mathcal{L}(\cdot, \cdot)$,

$$R(g) = \mathrm{E}_{\mathsf{w},\mathsf{z}} \{\mathcal{L}(\mathsf{z}, g(\mathsf{w}))\}. \tag{13}$$

The classifier $g_{\mathrm{B}}(\mathsf{w})$ that minimizes this risk is called “Bayes classifier” [14]. Using the 0/1-loss

$$\mathcal{L}(\mathsf{z}, g(\mathsf{w})) = \begin{cases} 0, & \text{if} \quad \mathsf{z} = g(\mathsf{w}) \\ 1, & \quad \text{otherwise,} \end{cases} \tag{14}$$

the Bayes classifier is identical with the *Maximum-A-Posteriori* (MAP) classifier, i. e., given a feature vector $\boldsymbol{w}$, the best choice for its label is the class with the highest *a-posteriori* probability

$$g_{\mathrm{B}}(\boldsymbol{w}) = g_{\mathrm{MAP}}(\boldsymbol{w}) = \underset{c_\ell}{\operatorname{argmax}} \{\mathrm{p}(\mathsf{z}{=}c_\ell | \mathsf{w}{=}\boldsymbol{w})\}. \tag{15}$$

Thus, many algorithms in statistical learning compute—explicitly or implicitly—based on a training data set $\mathcal{D}$ estimates $\hat{\mathrm{p}}_{\mathsf{z}|\mathsf{w}}(\mathsf{z} = c_q | \mathsf{w} = \boldsymbol{w})$, $q = 1, \ldots, Q$, for the *a-posteriori* probabilities and assign classes to inputs $\boldsymbol{w}$ based on these estimates.

3 Machine Learning for Autonomous Parking

The approach for autonomous parking which is presented in this work is based on a decomposition of a parking maneuver in sub-maneuvers that can be realized individually with an open-loop control in a vehicle. Such a decomposition of maneuvers is presented also in [1]. In [15] the authors introduce the term "dynamic primitives" for segments of trajectories. This decomposition into segments has similarities to the way how human drivers would plan a complex driving maneuver. In this approach the challenging task is the identification of transitions between sub-maneuvers. For this purpose, as shown in Figures 1 and 2, machine learning classifiers are used, having the task to detect those points in a parking maneuver where a transition from one sub-maneuver to the next one is necessary.

3.1 Ensemble Learning and GRBF

The classifiers that are used to detect the transition from one segment of a parking maneuver to the next one are constructed in a two-stage design process. In a first step, an ensemble learning algorithm, the *Random Forest* (RF), is used to find similarities between the feature vectors in the training data set and to generate a fair estimate of the generalization error related to the classification task and hereby of the complexity of the problem in hand. In the second step, the similarities between the feature vectors in the training data set are used for the generation of class-specific clusters, which are utilized for the construction of a *Generalized Radial Basis Function* (GRBF) classifier. Then, the trained GRBF-classifier can be implemented on a microcontroller with limited computational resources, while providing some additional advantages, e. g., a "fuzzy-like" interpretability of each decision.

3.1.1 Ensemble learning

Ensemble learning is a powerful machine learning technique that has proven a high performance in classification tasks on a variety of real-world applications. An ensemble can be characterized as a set of individual machine learning models that work in parallel and whose outputs are combined to the final output. One approach to explain why ensemble learning classifiers have a high generalization ability, i. e., according to Eq. (13) a low risk $R(g)$, is based on the bias-variance decomposition. In [16] it is shown that the bias-variance decomposition of the generalization error for classification problems does not lead to an additive relation between bias and variance, as it is the case in regression tasks, and that it is possible to reduce the classification error to its minimum value by reducing only the variance. If the decisions of the individual classifiers in an ensemble are combined using the majority vote scheme, this is a variance-reducing technique, which leads to a high generalization ability. The variance is reduced by a majority vote since uncorrelated errors made by the individual classifiers can be removed. Another explanation for the superior performance of ensemble learning classifiers compared to a "single best learner" is given in [17], where three reasons are mentioned: the training data might not provide sufficient information for choosing a single best classifier, the learning algorithms for the single best classifier might not be able to solve difficult search problems, and finally the hypothesis space for the single best classifier might not contain the Bayes classifier g_{B}.

3.1.2 RF Kernel

A well-known ensemble learning classifier is the RF algorithm. The RF algorithm has been introduced by Breiman in [18] and it is one of the most powerful known off-the-shelf machine learning procedures for classification. It is a randomized and aggregated version of the well-known [19] *Classification And Regression Tree* (CART) algorithm strengthened by the bagging (*b*ootstrap *agg*regat*ing*) technique. Given a RF with B trees, an intrinsic similarity measure in the input space between the input vectors $\boldsymbol{w}_m$ and $\boldsymbol{w}_n$ is given by the measure

$$prox(\boldsymbol{w}_m, \boldsymbol{w}_n) = \frac{B'(\boldsymbol{w}_m, \boldsymbol{w}_n)}{B}. \qquad (16)$$

Hereby, $B'(\boldsymbol{w}_m, \boldsymbol{w}_n)$ is the number of trees in the RF in which the input vector $\boldsymbol{w}_m$ and $\boldsymbol{w}_n$ lie in the same leaf. The measure $prox(\boldsymbol{w}_m, \boldsymbol{w}_n)$ has been introduced by Breiman in [18] as the proximity measure and it has been shown in [20] that it can be interpreted for full-grown CART-trees in the RF as a kernel

$$\kappa_{\mathrm{RF}}(\boldsymbol{w}_m, \boldsymbol{w}_n) = prox(\boldsymbol{w}_m, \boldsymbol{w}_n). \qquad (17)$$

Besides providing a similarity measure, that will be used for the construction of GRBF-classifiers (see next Subsection), the RF-algorithm also offers the possibility to generate an unbiased estimate of the generalization error if the number B of trees in the

RF is large [18]. This is realized by the bootstrap procedure when training the trees in the RF, such that each training pattern is not seen by approximately 36% of the B trees in the RF. By evaluating each pattern with a majority vote only among those trees which have not seen this pattern during training one obtains a so-called out-of-bag estimate of the generalization error. In [21] Breiman gives empirical evidence that the out-of-bag estimate is as accurate as using a test set of the same size as the training set. This way by using a RF classifier g_{RF} it is possible to determine a good estimate $R_{\mathrm{oob}}(g_{\mathrm{RF}})$ of the risk $R(g_{\mathrm{RF}})$, which is also a measure of the complexity of the classification task that must be solved.

3.1.3 GRBF

The design of GRBF in this work is based on [20], where the main idea is to carry over the good generalization properties of the RF algorithm to GRBF classifiers by approximating the RF-kernels with Gaussian kernels. After training a RF classifier g_{RF} and computing the kernel $\kappa_{\mathrm{RF}}(\boldsymbol{w}_m, \boldsymbol{w}_n)$ between each pair of feature vectors $(\boldsymbol{w}_m, \boldsymbol{w}_n)$ from the training set a class-wise clustering of the training data is performed. This leads to H clusters and for each cluster the sample mean $\boldsymbol{t}_h$ and the sample covariance matrix $\boldsymbol{C}_h$ are computed. For a feature vector $\mathbf{w}$ the similarity $\mathsf{s}_h(\mathbf{w}, \boldsymbol{t}_h)$ to the sample mean $\boldsymbol{t}_h$, which can be interpreted as a template of the data in the h-th cluster, can be implemented by

$$\mathsf{s}_h(\mathbf{w}, \boldsymbol{t}_h) = \exp\left(-\gamma(\mathbf{w}-\boldsymbol{t}_h)^{\mathrm{T}} \boldsymbol{C}_h^{-1}(\mathbf{w}-\boldsymbol{t}_h)\right), \quad (18)$$

where $\gamma > 0$ is a tuning parameter. The GRBF classifier is realized using the architecture presented in Figure 7.

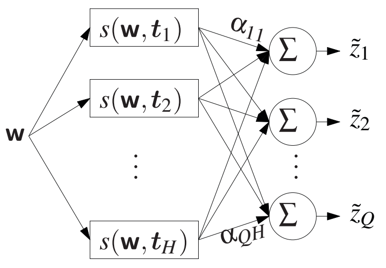


Figure 7. Architecture of a GRBF classifier

The output $\tilde{\mathbf{z}} = [\tilde{\mathsf{z}}_1, \ldots, \tilde{\mathsf{z}}_Q]^{\mathrm{T}}$ of the GRBF classifier is computed as

$$\tilde{\mathbf{z}} = \boldsymbol{A}\mathbf{s}, \quad (19)$$

where $\boldsymbol{A} \in \mathbb{R}^{Q \times H}$ is a weight matrix with the (q,h)-th entry being $\alpha_{q,h}$ and $\mathbf{s} = [\mathsf{s}_1(\mathbf{w}, \boldsymbol{t}_1), \ldots, \mathsf{s}_H(\mathbf{w}, \boldsymbol{t}_H)]^{\mathrm{T}}$ denotes the similarity of the feature vector $\mathbf{w}$ to the H centers. Having M patterns in the training data set the weights $\alpha_{q,h}$ and the parameter γ are computed by the minimization of

$$\varepsilon = \frac{1}{M} \sum_{m=1}^{M} ||\tilde{\boldsymbol{z}}_m - \boldsymbol{e}_{q(m)}||^2, \quad (20)$$

with $\boldsymbol{e}_{q(m)}$ being the unit vector representation of the class c_q belonging to the feature vector $\boldsymbol{w}_m$ and $\tilde{\boldsymbol{z}}_m = \boldsymbol{A}\mathbf{s}_m$, where $\boldsymbol{s}_m \in \mathbb{R}^H$ denotes the similarities of $\boldsymbol{w}_m$ to the H templates $\boldsymbol{t}_h$. Finally, an output $\tilde{\mathbf{z}} \in \mathbb{R}^Q$ can be transformed by a mapping m

$$m : \mathbb{R}^Q \to [0,1]^Q, \quad \tilde{\mathbf{z}} \mapsto \hat{\mathbf{z}} \quad (21)$$

as described in [22] to a vector $\hat{\mathbf{z}}$ of estimated *a-posteriori* probabilities

$$\begin{aligned} \hat{\mathbf{z}} &= [\hat{\mathsf{z}}_1, \ldots, \hat{\mathsf{z}}_Q]^{\mathrm{T}} & (22) \\ &= [\hat{\mathsf{p}}_{\mathsf{z}|\mathbf{w}}(\mathsf{z}=c_1|\mathbf{w}=\boldsymbol{w}), \ldots, \hat{\mathsf{p}}_{\mathsf{z}|\mathbf{w}}(\mathsf{z}=c_Q|\mathbf{w}=\boldsymbol{w})]^{\mathrm{T}}. \end{aligned}$$

This way by choosing for a given feature vector $\boldsymbol{w}$ as label the class with the highest estimated *a-posteriori* probability an estimate of the MAP-classifier from Eq. (15) is realized.

The "fuzzy-like" interpretability that can be obtained with the GRBF classifier must be understood as a statement of the form: "The feature vector $\mathbf{w}$ obtains the class label c_q because it is close to at least one center $\boldsymbol{t}_h$ that belongs to class c_q and far away from the centers that belong to the other classes". Besides the good generalization performance that is inherited from the RF algorithm and the fuzzy-like interpretability, these GRBF classifiers require low computational resources making them attractive for implementations on microcontrollers.

3.2 Data Generation

In order to train the classifiers to detect various segmentation points, several training data sets are needed. For this reason, a simulation program was designed with the purpose of creating them. Given a certain range for the initial conditions, this program is able to find the right maneuver that drives the vehicle in the desired parking spot, and store it in a database. This is done in two different ways, mainly depending on the geometry of the considered parking area (i. e., parallel or cross parking), but always applying the same concept. The maneuver is splitted into segments which can be either straight lines or arcs, whose lengths depend on the initial state of the car and on its relative position with respect to the parking spot (see Figures 8 and 9). The simulation program changes iteratively the lengths of the segments until the vehicle is able to enter its parking spot without occupying the adjacent ones. By this procedure a labeling of the feature vectors is performed.

For the simulation the geometric and physical characteristics of the small-scale vehicle which will be employed in the testing phase are used.

3.2.1 Parallel-Parking

The maneuver structure for the parallel parking is depicted in Figure 8. Two segmentation points that divide the maneuver into three segments can be seen: the first is straight, the second and the third are driven at the maximum steering angle value, respectively clockwise and counterclockwise. The length of these segments, or their duration given a certain vehicle speed, is to be found by means of the algorithm mentioned in Subsection 3.2. In this way, the parking area and the vehicle state can be mapped to a feature space. The feature spaces of the classifiers used for the maneuvers are shown later in Figures 13a and 13b.

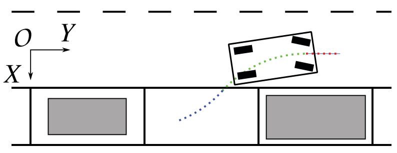


Figure 8. Parallel parking maneuver

3.2.2 Cross-Parking

In Figure 9 the structure of the cross parking maneuver is depicted. As in the case of the parallel parking, it is divided into three segments, which differ from the previous ones only in their relative position. The second segment here is obtained driving with maximum value of the steering angle counterclockwise, while the third one is driven clockwise. Similar to the previous case, also the feature spaces of the classifiers used for the cross parking maneuver are shown later in Figures 13c and 13d.

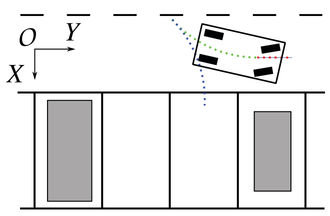


Figure 9. Cross parking maneuver

3.2.3 Oblique-Parking

Figure 10 visualizes the oblique parking, where the maneuver is made up of only two segments. Therefore, only one segmentation point is defined. The first segment is straight, while the second the second is driven at the maximum steering angle value. Also here, the length of these segments will be found using the procedure explained in Subsection 3.2.

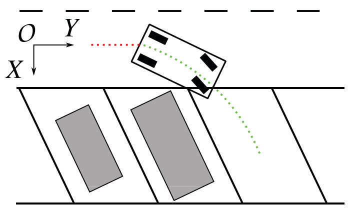


Figure 10. Oblique-parking maneuver

3.3 Feature Spaces and Visualization

The generation of the features, which is represented in Figures 1 and 6 by the block “FEATURE GENERATION”, is described in this Subsection. For the classification tasks that must be performed for the segmentation of parking maneuvers three-dimensional feature spaces are chosen, i. e., $\mathbf{w} \in \mathbb{R}^3$. Being able to work with three-dimensional feature spaces has the huge advantage that the decisions of the classifiers can be visualized and hereby validated by experts. The three features are the x distance d_x, the y distance d_y, and the yaw angle ψ. While the yaw angle has a clear meaning (namely the orientation of the car in the global reference frame as defined in Figure 4), the two distances need to be clarified. They are the oriented projections on the axes of the global reference frame of the vector that goes from the center of gravity of the vehicle to the most critical point in the parking area. In the parallel parking scenario this point is the rear left corner of the rectangle representing the parking spot in front of the target spot. In the cross parking scenario the critical point is the rear right corner of the rectangle representing the parking spot on the right of the target spot. The bold vectors in Figures 11 and 12 visualize how the features d_x and d_y are computed.

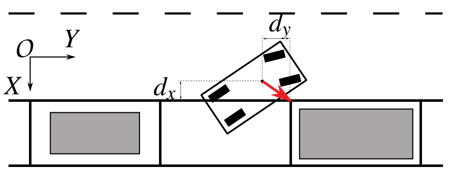


Figure 11. Critical point in the parallel parking scenario

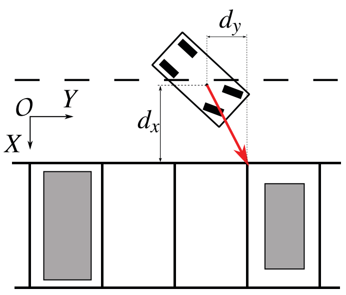
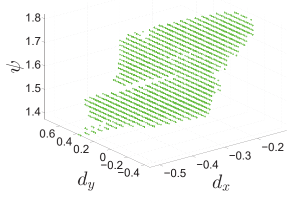
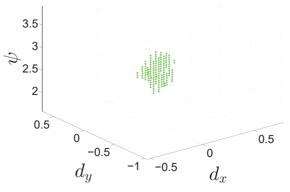


Figure 12. Critical point in the cross parking scenario

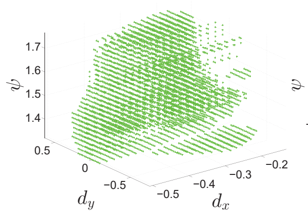
Figures 13a and 13b show the feature spaces of the two classifiers that are used for the segmentation of the parallel parking maneuver. Being only three-dimensional, the feature space has been sampled for the generation of the figures and only those feature vectors where the classifiers detect a transition from one sub-maneuver to the next are represented with markers. These markers form the “clouds” in the figures. Similarly for the cross parking maneuver the feature spaces and the decisions of the two classifiers are visualized in Figures 13c and 13d.



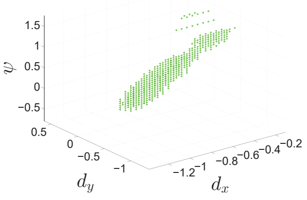
(a) Classifier for the first segmentation point of the parallel parking maneuver



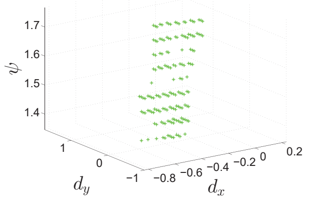
(b) Classifier for the second segmentation point of the parallel parking maneuver



(c) Classifier for the first segmentation point of the cross parking maneuver



(d) Classifier for the second segmentation point of the cross parking maneuver



(e) Classifier for the segmentation point of the oblique parking maneuver

Figure 13. Feature spaces and decisions of the four classifiers

The visualization of the decision boundaries of the four classifiers reveal that their learned behavior conforms with expert knowledge.

4 Validation Using a Small-Scale Autonomous Vehicle

In order to test the parking maneuver algorithms, a model of a fully autonomous car was used. This has been provided by Audi AG for its contest *Audi Autonomous Driving Cup*.

4.1 Hard- and Software

In Figure 14 there is a scheme of the hardware of the small-scale car used for the presented tests.

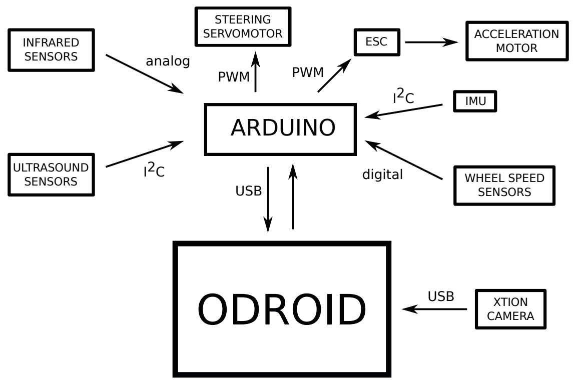


Figure 14. Hardware of the small-scale car used for testing

The core of the computing system is an ODROID-X2 board. It is used for the main signal processing, sensor funsion, situation interpretation and action planning. This board is connected via USB with an Arduino Due board which takes care of the interface with sensors and actuators, i. e., reading the values of the signals coming from the sensors and writing the calculated values to the actuators or the actuators controllers. An ASUS xtion camera is directly connected, also via USB, to the ODROID board. This is able to provide both RGB images and depth maps by the aid of a built-in infrared device. Other exteroceptive sensors are the infrared and the ultrasound sensors, which are homogeneously distributed around the vehicle.
As far as the software is concerned, on the ODROID board, a Linux distribution is installed. The framework in which all the functions have been integrated is ADTF (Automotive Data and Time-Triggered Framework), that is a framework widely used in the automotive industry. All the functions have been written in C++, making use of OpenCV libraries for the basic image processing functions.

4.2 Validation Procedure and Results

The size of the training data for the four classifiers is presented in Table 1.

The out-of-bag generalization errors of the trained RF-classifiers (see Subsection 3.1) and the resubstitution errors of the GRBF-classifer are presented in Table 2, 3 and 4.

Table 1. Number of patterns in the training data set for the parallel, cross and oblique parking maneuver

	Parallel	**Cross**	**Oblique**
Classifier 1	1512	1728	7560
Classifier 2	2354	2514	—

Table 2. Performance of Classifiers for the Parallel Parking Maneuver

	Classif. 1	**Classif. 2**
OOB error RF	6.18 %	1.32 %
Resubstitution error GRBF	3.44 %	1.15 %

Table 3. Performance of Classifiers for the Cross Parking Maneuver

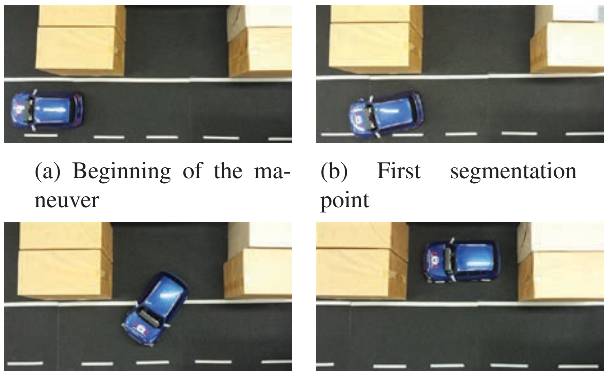
	Classif. 1	**Classif. 2**
OOB error RF	8.12 %	1.79 %
Resubstitution error GRBF	8.85 %	1.63 %

Table 4. Performance of Classifier for the Oblique Parking Maneuver

	Classif. 1
OOB error RF	1.23 %
Resubstitution error GRBF	2.46 %

As expected from the visualization in Figure 13 the classifiers trained to detect the first segmentation point perform worse than the classifiers for the second segmentation point for both parallel and cross parking maneuvers. In particular the detection of the first segmentation point in cross parking maneuvers is a hard classification task.
Tables 5 and 6 report the results of a series of tests carried on with the small-scale autonomous vehicle in order to validate the whole parking procedure and compare it to the simulation results. Figures 15a

to 15d show prototypically an autonomous parallel parking maneuver that was implemented using the GRBF-classifiers.



(a) Beginning of the maneuver (b) First segmentation point

(c) Second segmentation point (d) Final position of the car

Figure 15. Parallel parking maneuver tested on the car model

A maneuver is considered to be successful when the final position and orientation of the car are within the same boundaries used for the training procedure. This is given when the center of gravity of the car is inside a square centered at the center of the parking spot, and its orientation, before alignment, does not differ more than 45° from the ideal one. The cases 1, 2, and 3 in Tables 5 and 6 (first, second, and third row) correspond to those tests in which the position and the orientation of the vehicle at the beginning of the maneuver are: *inside* the region of the feature space where the classifiers were trained (case 1), *outside* of this region but *within* 130 % of its boundary (case 2), *outside* of this region and *between* 130 % *and* 160 % of its boundary (case 3). These tests were performed exemplarily for the parallel parking maneuver.

- For the first case the tests show that all maneuvers succeeded bringing the vehicle in the right final position.
- In the second case, even though the initial conditions of the vehicle are outside the training-region, 60 % of the maneuvers succeeded and, when they failed, the two classifiers have the same percentage of error.
- In the third case, all the 19 failing maneuvers were already due to the first classifier, for the first segmentation point. In this case the outputs of the second classifier are ignored due to the logic in the block "DECISION SUB-MANEUVER" in Figure 2.

The percentages in Table 6 are relative to the number of failed tests for each case.

Table 5. Results of 100 tests carried on with the small-scale autonomous vehicle (successful maneuvers)

	Completed successfully
Case 1 (40 tests)	40
Case 2 (30 tests)	18
Case 3 (30 tests)	11

Table 6. Results of 100 tests carried on with the small-scale autonomous vehicle (most failing classifier)

	Most failing classifier
Case 1 (40 tests)	—
Case 2 (30 tests)	Classifier 1 and 2 (50%)
Case 3 (30 tests)	Classifier 1 (100%)

Since the region of the feature space covered by the training data is the most frequently expected, the focus should lie on case 1. The out-of-bag error of the RF classifiers indicates a generalization error of approximately 6 % which for 40 tests means 2 failures. From the resubstitution error of the GRBF classifiers (3.44 %), which in general is smaller than the true generalization error, we could expect at least 1 failure. In the tests no failure was observed for case 1, which is due to the small number of trials.

5 Conclusion

A machine learning approach to realize an autonomous parking maneuver has been introduced in this work. The approach relies on dividing a complex driving maneuver into segments. The detection of the transition points from one segment to another is performed with GRBF-classifiers which were generated by using the kernel of an ensemble classifier, the RF algorithm. The training and a first validation have been done using simulation data. Then, the classifiers were implemented into a small-scale autonomous vehicle and the whole procedure for autonomous parking was validated successfully in a number of trials. Besides the high robustness, which has been pointed out in previous Section, the presented method does not require much computational effort both for the sensor data acquisition and pre-processing, and also for the computation of suitable parking maneuvers. These are big advantages of the presented parking method over different currently-adopted strategies mentioned in Section 1. They can be exploited to reduce both the parking maneuver computational time and the complexity of the whole hard- and software system architecture. The results demonstrate that the methodology of training classifiers using only simulation data and then implementing them in real systems is extremely well-suited for this application.

Acknowledgment

The authors would like to thank the committee of the Audi Autonomous Driving Cup for providing the small-scale autonomous vehicle.

References

[1] K. Min, J. Choi, H. Kim, and H. Myung, Design and implementation of path generation algorithm for controlling autonomous driving and parking, in 12th International Conference on Control, Automation and Systems (ICCAS), Jeju Island, Korea, 2012, pp. 956–959

[2] I. E. Paromtchik and C. Laugier, Autonomous parallel parking of a nonholonomic vehicle, in Proc. of the IEEE Intelligent Vehicles Symposium, Tokyo, Japan, 1996, pp. 13–18

[3] J. Z. Kolter, C. Plagemann, D. T. Jackson, A. Y. Ng, and S. Thrun, A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving, in IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA, 2010, pp. 839–845

[4] T. K. Lau, Learning autonomous drift parking from one demonstration, in Proc. of the IEEE International Conference on Robotics and Biomimetics, Phuket, Thailand, 2011, pp. 1456–1461

[5] M. R. Heinen, F. S. Osório, F. J. Heinen, and C. Kelber, Seva3d: Using artificial neural networks to autonomous vehicle parking control, in 2006 International Joint Conference on Neural Networks, Vancouver, BC, Canada, 2006, pp. 4704–4711

[6] N. N. Samani, J. Ghaisari, and M. Danesh, 2nd international econference on computer and knowledge engineering (iccke), in 2006 International Joint Conference on Neural Networks, 2012, pp. 117–122

[7] D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg, Design of radial basis function-based controller for autonomous parking of wheeled vehicles, in Proc. of the American Control Conference, Baltimore, Maryland, 1994, pp. 806–810

[8] S. Kim, W. Liu, and K. A. Marczuk, Autonomous parking from a random drop point, in 2014 IEEE Intelligent Vehicles Symposium (IV), Dearborn, Michigan, USA, 2014, pp. 498–503

[9] Y. Wang and X. Zhu, Hybrid fuzzy logic controller for optimized autonomous parking, in American Control Conference (ACC), Washington, DC, USA, 2013, pp. 182–187

[10] Z. Joung, K. J. W. X. DongJi, and K. Y. Bae, A study of autonomous parking for a 4-wheel driven mobile robot, in Proc. of the 26th Chinese Control Conference, Zhangjiajie, Hunan, China, 2007, pp. 179–184

[11] R. Schubert, E. Richter, and G. Wanielik, Comparison and evaluation of advanced motion models for vehicle tracking, in 11th International Conference on Information Fusion, 2008

[12] B. Siciliano and O. Kathib, Springer Handbook of Robotics, Berlin, Germany: Springer, 2008

[13] P. S. Maybeck, Stochastic Models, Estimation and Control, New York, NY, USA: Academic Press, 1979, vol. 1

[14] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer-Verlag, 2001

[15] P. Banerjee and R. Nevatia, Dynamics based trajectory segmentation for uav videos, in Conference on Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International, Boston, MA: IEEE, 2010, pp. 345 – 352

[16] J. H. Friedman, On bias, variance, 0/1-loss and the curse of dimensionality, Data Mining and Knowledge Discovery, vol. 1, pp. 55–77, 1997

[17] T. G. Dietterich, Machine learning research: Four current directions, AI Magazine, vol. 18, no. 4, pp. 97–136, 1997

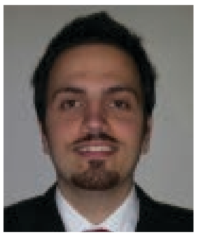
[18] L. Breiman, Random forests, Machine Learning, vol. 45, no. 1, pp. 5–32, 2001

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees, ser. The Wadsworth & Brooks/Cole Statistics/Probability Series, Wadsworth, 1984

[20] M. Botsch and J. A. Nossek, Construction of interpretable radial basis function classifiers based on the random forest kernel, in IEEE World Congress on Computational Intelligence 2008 (WCCI 2008), 2008

[21] L. Breiman, Out-of-bag estimation, University of California, Berkeley, Tech. Rep., 1996

[22] J. Schürmann, Pattern Classification: a unified view of statistical and neural approaches, John Wiley & Sons, 1996



Gennaro Notomista received the Bachelor's Degree in Mechanical Engineering in 2012 from the University of Naples "Federico II", Naples, Italy, a Master of Engineering in Automotive Engineering in 2015 from the Technische Hochschule Ingolstadt, Germany, and a Master of Science in Mechanical Engineering in 2016 from the University of Naples "Federico II", all with honors. From 2016 he is a PhD student in Robotics at the Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA, USA. He is a student member of IEEE.

Michael Botsch received the diploma and doctoral degrees in electrical engineering, both with honors, from Technische Universität München, München, Germany, in 2005 and 2009. He worked for five years in the automotive industry as Development Engineer at Audi AG in the field of active safety systems. In October 2013 he was appointed Professor for Vehicle Safety and Signal Processing at Technische Hochschule Ingolstadt in the Department of Electrical Engineering and Computer Science. He is the Associate Scientific Director of the vehicle safety research center CARISSMA at Technische Hochschule Ingolstadt. His research interests are in signal processing and automotive applications. Prof. Botsch is a member of IEEE and VDE.