High-Resolution Radar Based Object Classification for Automotive Applications

Ganesh Nageswaran
Department of Mechanical and Process Engineering
Technical University of Kaiserslautern
Kaiserslautern, Germany
2016
gechuvittal@gmail.com

***Abstract*—This paper presents a method for classifying traffic participants based on high-resolution automotive radar sensors for autonomous driving applications. The major classes of traffic participants addressed in this work are pedestrians, bicyclists and passenger cars. The preprocessed radar detections are first segmented into distinct clusters using density-based spatial clustering of applications with noise (DBSCAN) algorithm. Each cluster of detections would typically have different properties based on the respective characteristics of the object that they originated from. Therefore, sixteen distinct features based on radar detections, that are suitable for separating pedestrians, bicyclists and passenger car categories are selected and extracted for each of the cluster. A support vector machine (SVM) classifier is constructed, trained and parametrised for distinguishing the road users based on the extracted features. Experiments are conducted to analyse the classification performance of the proposed method on real data.**

***Index Terms*—radar, clustering, feature extraction, object classification, machine learning, autonomous driving**

I. Introduction

High-resolution radars in the frequency range 77-79 GHz are recently developed for usage in many automotive applications. With an increased resolution, multiple reflection points are detected from an extended object. Therefore the usual assumption of an object as point target is not valid any more and the spatial extent of the object also needs to be considered. Moreover, every object has it's own distinct features. For example, a pedestrian walks with a velocity which is very distinct from the velocity of a car, or the dimensions of a cyclist are smaller compared to a truck. Extracting such features out of measurements originating from an object would help to classify that object.

Vast number of methods have already been proposed for detecting and classifying objects based on range sensors. In a method proposed in [1], [2], objects can be detected and classified based on shape models. Classification based on measurement characteristics and motion analysis is presented in [3], [4]. Distance based clustering of detections and fitting one of the pre-defined shape models to the cluster to infer the object type is proposed in [5]. [6] proposes a method of joint object tracking and classification based on laser scanner, where the defined object classes are 0-axis, 1-axis or 2-axes objects. A Markov chain based on different observation positions of the object is constructed. The transition probabilities are defined based on the object manuever and the corresponding observation of the sensor. In [7], a method for segmentation of measurements from laser scanner and consecutive feature extraction from these segments is proposed. The extracted features are then used for training different machine learning based classifiers to detect pedestrians. As the measurement resolution of radar sensors in automotive domain have significantly improved in the recent years, we propose an hybrid approach in this work, which combines the clustering algorithms typically applied for laser scanner sensors, with the machine learning techniques for classifying road users.

The proposed solution constitutes a sequence of steps which can be abstractly put as segmentation, feature extraction and classification. After an initial pre-processing, density based spatial clustering of applications with noise (DBSCAN) algorithm [8] is used for segmenting high-resolution radar detections into distinct clusters. From the object tracking point of view, each cluster is fitted with pre-defined geometric shape and a reference point is chosen accordingly, which could be used as a reference measurement point of the object for tracking. Apart from the geometric shape, sixteen pre-defined features that could explain the potential characteristics of the underlying object are extracted for each of the cluster. A support vector machine (SVM) classifier is constructed and trained with the help of these object features, which can then be used in real-time for classifying pedestrians, bicyclists and passenger cars.

The remainder of this article is organised as follows. Section II presents the methods for segmentation of the radar data, which involves the pre-processing and data clustering steps. Different geometric and characteristic features that could be helpful in distinguishing the road users and how they can be extracted from radar data are presented in section III. Classification based on SVM classifier and the results are presented in section IV and section V. A short summary with remarks is given in section VI.

This work was carried out by the author during his tenure as Scientific Employee at the Department of Mechanical and Process Engineering at Technical University Kaiserslautern, in cooperation with the company Knorr-Bremse System for Commercial Vehicles Private Limited in Schwieberdingen.

II. Segmentation of radar data

A. Data pre-processing

Radar measures the range, azimuth and Doppler velocity, which are extracted for each measurement point depending on the underlying measurement principle, for example with frequency modulated continuous wave (FMCW) detection method. Unlike a point target, multiple detections can originate from an extended object. Therefore the segmentation algorithm tries to sort out the detection points as unique groups. Ideally each group or cluster should contain all the detections that belong to a single extended object. A typical problem with radar sensor is the so called double reflection, where the electro-magnetic waves from the radar reflected by the object surface is strong enough to be reflected back again by the radar mounting or the ego-vehicle surface. If the amplitude is high enough, these double reflections would create a peak in the Fast Fourier Transform (FFT), thereby creating false positive objects. Therefore, some pre-preprocessing of the detections is required in order to avoid these false positive objects. The double reflections from the same object need to filtered out, before grouping them. Although the measured azimuth in the case of a double reflection would be nearly the same, the range and Doppler velocity would have values that are in multiples of two or more.

$$\begin{aligned} \theta_{double} &\approx \theta_{single} \\ r_{double} &= 2 \cdot r_{single} \\ \dot{r}_{double} &= 2 \cdot \dot{r}_{single} \end{aligned} \tag{1}$$

Additionally, also the detection points which have an unrealistic Doppler velocity values are filtered out before clustering. These are typically clutter measurements which is prevalent with radar sensors.

B. Clustering of radar detections

After the pre-processing of the raw detections, the remaining detection points can be understood as belonging to the objects in the environment, either pedestrians, bicyclists, vehicles or other infrastructures. The basic concept of the DBSCAN algorithm is that, any point in a cluster should contain a certain number of other points within a particular defined radius. Which means, a group of points make a cluster if their density is higher than particular threshold. The distance between two points p and q can basically be described by any distance function $dist(p, q)$. Euclidean distance function is used in this work. Basic definitions are made in [8] which defines *Eps*, *MinPts*, *core point* and *border points*.

Eps is defined as the threshold radius, which represents the neighbourhood of a point. The *Eps-neighbourhood* of a point p denoted as $\mathrm{N}_{Eps}(p)$ is defined by $\mathrm{N}_{Eps}(p) = [q \in D | dist(p, q) \leq Eps]$. Further as illustrated in the Figure 1, the point p is a *core point*, if it has at least *MinPts* number of points in it's neighbourhood and also those neighbouring points are *directly density-reachable* from p. If there is a point q_1 within the Eps-neighbourhood of p but do not have *MinPts* number of points that are directly density-reachable from it, then q_1 would be a border point. q_1 is directly density-reachable from p but not the other way, as directly density-reachable is defined to be symmetric only for the connections between the core points. Considering another point r in the Figure 1, q_1 is said to be *density-reachable* from r through the path of points $p_1, \ldots p_n$, with $p_1 = r$, $p_n = q_1$, and rest of the points p_{i+1} also being core points. The definition of *density-connected* connects two border points q_2 and q_1 through the point r as they are both density-reachable from the core point r. Now starting from point p, including all the core points that directly density-reachable from p and the border points q_1 and q_2 that are density-connected between them and density-reachable from p form a cluster.

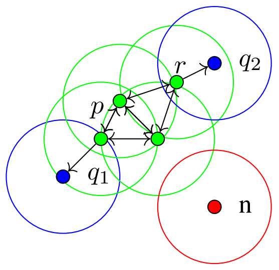


Fig. 1: Illustration of the DBSCAN algorithm

Clustering parameters selection: The parameters *Eps* and *MinPts* usually could to be selected based on the application at hand. [8] proposes *k-dist* graph method for selection of these parameters, from the thinnest cluster of points. However, the suggestion is intended mainly for unknown application situations. For our application case, the objects of interest in the environment are more or less known a-priori, which of all the pedestrian category has the least spatial extension. Another factor in selecting the parameter is that, the density and number of points reflected from an object depends strongly on the position of the object with respect to sensor. When the object is closer to the radar sensor it generates more reflection points, whereas when it is detected at the farthest position in the FoV, only a single point could be generated. From many measurements it was empirically observed that all the object classes generated as less as one detection point at the edge of FoV. At closer positions to the sensor, pedestrians generated up to six detections, bicyclists up to seven detection points and a passenger car up to twenty detection points. Intuitively, for many low speed use cases of autonomous driving, the area very close to the ego-vehicle is of high relevance and the clustering in that area needs to be reliable, than at a higher distance.

III. Feature extraction

The ideal output of the above defined clustering are well defined distinct clusters, each representing a distinct object. Further in the feature extraction step, distinct object hypotheses are searched and extracted for each segment. On an abstract level, the hypotheses can be divided into two categories, namely the shape or extension of the object and the distinct characteristic features of the object. Pedestrians are considered in this work to have a circular extension, bicyclists stick extension and other vehicles as rectangular boxes. Therefore three shape model assumptions are made with O-shape representing pedestrians, I-shape representing one side of vehicle and bicyclist and L-shape representing two sides of vehicle. Similar kind of approach is used in many of the earlier works such as [5]. The parameters of the shape feature are then extracted by fitting the detection points in each cluster to one of the above defined models sequentially.

A. *Fitting a geometric shape*

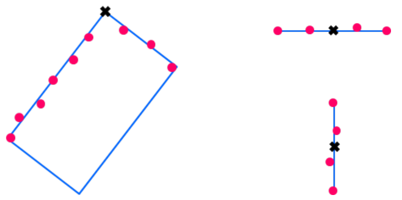
Before beginning with the actual shape fitting, an idea on what kind of shapes should be tried for each cluster can be obtained by checking the 2-dimensional spatial spread of the detections in the cluster. In other words, if the detections are spread along both the longitudinal and lateral axes, and there are many detections available in the cluster, the cluster would supposedly have a larger extension. If the spread is only along one axis, it would potentially be a line type. A method for applying such a pre-selection is proposed in [5]. Principal component analysis (PCA) is applied on each of the clusters and the corresponding eigenvalues of the covariance matrix are calculated. If the ratio of the eigenvalues is greater than a certain threshold, it would mean that the spread is dominated by one of the axes and therefore only I-shape is tried further for that cluster. In the other case, when the ratio of the eigenvalues is lesser than the pre-defined threshold, the spread is potentially along both the axes and therefore L-shape is tried to be fit. Additionally, if the number of detection points in the cluster is less than N_{min}, the cluster is assumed to have O-shape.

There are many methods and algorithms available from the field of robotics for extracting lines from a set of points. Some important methods are summarized in [9]. If the model pre-selection denotes that a segment must be tested for both L-shape and I-shape, iterative end point fit (IEPF) algorithm is first applied on that segment. The algorithm is also known in literature under the variants Ramer-Douglas-Peucker (RDP) algorithm or split-and-merge algorithm. The basic idea is to fit a line with first and the last point of the segment and find a point which has the maximum orthogonal distance to this line. This point intuitively is the corner point and the set is split into two individual set of points at this point. Lines are then fit individually to these sets of points. Now we are left only with the segments that need to be looked for I-shape.

Hough-Transform introduced by Richard Duda and Peter Hart is a widely used method in the field of computer vision for extracting line segments out of images. The key idea is to represent a line in it's Hesse normal form

$$r = x\cos\theta + y\sin\theta \tag{2}$$

where r is the perpendicular distance from origin to the line and θ the angle made by the line with x-axis. The line is represented in the above normal form instead of general form $y = mx+b$ because in case of vertical line the slope parameter



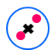


Fig. 2: Considered geometric shapes for detection clusters. The black color cross represents the corresponding reference point (RP) chosen for the fitted shape

would lead to singularity. Whereas a vertical line in normal form can be represented with $\theta = 90°$and r equal to the x-axis intercept. The parameter space or Hough space is given by (r, θ). Generally speaking, any number of lines can be drawn through a given point in xy-plane. This would mean a sinusoidal curve for each point on the Hough-space with varying θ and r. Such sinusoidal segment is constructed for all the points in the cluster. Then the value of r' and θ' in the Hough-space, where the sinusoidal curves of all points intersect would give the line in the xy-plane, which passes through all those points or in other words a line which fits the points in the cluster.

After fitting a line to the points, the clusters which showed a possibility of L-shape are checked for line segments which are almost perpendicular to each other. In that case, the line segments are merged to fit the L-shape and a rectangular bounding box covering the nearest and farthest coordinate in the cluster is overlayed on the cluster as represented in Figure 2 to represent it's shape hypothesis. The orientation of the cluster is calculated by PCA. The covariance and eigenvectors are calculated similar to the method as described previously for pre-selection. The orientation of the cluster is then given by the direction of the eigenvector $\mathbf{v_1}$ with highest eigenvalue λ_1

$$\psi_L = \tan^{-1}\left(\frac{\mathbf{v_1}(y)}{\mathbf{v_1}(x)}\right) \tag{3}$$

In order to calculate the four corner points of the bounding box, all the points in the cluster are first rotated by the orientation angle, so that the eigenvector $\mathbf{v_1}$ is parallel to the x-axis. After rotation, the maximum and minimum values of the x and y coordinates in the cluster represent the length and width of the bounding box. The coordinates of the corner points (p_1, p_2, p_3, p_4) representing the bounding box are finally calculated by again rotating the points of the cluster by it's orientation angle in the opposite direction. The corner point of the bounding box closest to the sensor can be then chosen as reference measurement point for the cluster, which could be used by the tracking algorithms. If there are no perpendicular line segments that are to be merged, the cluster is then denoted with I-shape hypothesis and the middle point of the line is chosen as the reference point. The orientation of the I-shape is then same as the estimated angle θ'. As stated before, if there are not enough points for both of the L-shape and I-

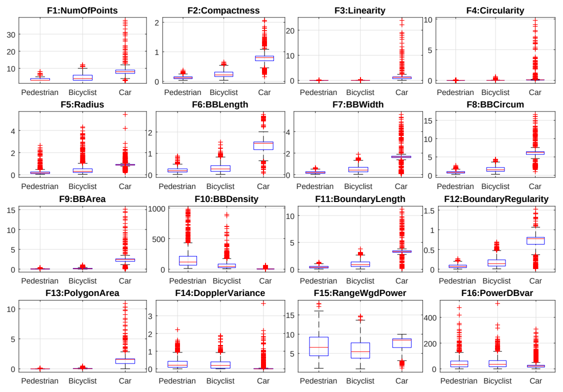


Fig. 3: Boxplot of all the features for each class

shape, the cluster is denoted with O-shape. In this case, the orientation is not known and set to zero, and the point can be chosen as reference measurement point.

B. Characteristic feature description

In addition to the geometric features, each cluster basically contains other characteristics features as well, which are representative of the actual object class that the cluster belongs to. Methods for extraction of characteristic features from measurement clusters and how these features can be used to classify objects has been previously studied in [10], [11] and [12], to name a few. In addition to the features described in [10], basically meant for laser scanner, new radar specific features are defined and extracted for each of the cluster in this work. The features that are of interest for each cluster C of radar detections are:

- *Number of points:* denotes the total number of detection points in that cluster

$$NumOfPoints = |C_i| \tag{4}$$

- *Standard deviation:* The standard deviation of the position of points is given by

$$Compactness = \sqrt{\frac{1}{n-1}\sum_j \|\mathrm{x_j} - \bar{\mathrm{x}}\|^2} \tag{5}$$

  where $\bar{\mathrm{x}}$ is the centroid of the cluster and $\mathrm{x_j}$ the positions of each point.
- *Linearity:* This feature denotes the residual sum of squares between the cluster points and a line fitted to the cluster points [10]. Given the line parameters $r^{'}$ and $\theta^{'}$, linearity can be calculated as

$$Linearity = \sum_j (x_j \cos(\theta') + y_j \sin\theta' - r')^2 \tag{6}$$

- *Circularity:* The residual of sum of squares to a circle fitted for points in the cluster is denoted by this feature. If a circle of radius R and center (x_c, y_c) is fitted to the points by least squares method, then the circularity of the cluster is given as in [10] by

$$Circularity = \sum_{j=1}^{n}\left(R - \sqrt{(x_c - x_j)^2 + (y_c - y_j)^2}\right)^2 \tag{7}$$

  with (x_j, y_j) denoting the positions of detection points in Cartesian coordinates.
- *Radius:* Denotes the radius R of the circle fitted to the cluster for calculating circularity feature.
- *Length of minimal bounding box:* The length of a bounding box $BBLength$ that would cover the points in the cluster is calculated as explained in section III-A. For perfect line this value would be zero.
- *Width of minimal bounding box:* This feature represents the width of the assumed bounding box $BBWidth$, again as calculated in section III-A.
- *Circumference of assumed bounding box:* Circumference of the bounding box

$$BBCircum = 2 \cdot (BBLength + BBWidth) \tag{8}$$

- *Area of assumed bounding box:* The area of the rectangular bounding box is calculated as

$$BBArea = BBLength \cdot BBWidth \tag{9}$$

- *Density of assumed bounding box:* Density of the bounding box is the ratio of number of points in the cluster n to its area $area_{BB}$

$$BBDensity = \frac{NumOfPoints}{BBArea} \tag{10}$$

- *Length of the boundary:* The length of the contour formed by connecting the points in a cluster is calculated by

$$BoundaryLength = \sum_j d(p_j, p_{j-1})$$
$$d(p_j, p_{j-1}) = \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2} \tag{11}$$

- *Boundary regularity:* The standard deviation of the distances $d(p_j, p_{j-1})$ as in [10].
- *Polygon Area:* The area of the polygon fitted to the cluster of detection points.
- *Doppler variance:* This feature denotes the variance of the Doppler velocity within a cluster. Due to the movement of the arms and legs, the Doppler variance of the pedestrian class is expected to be higher than other classes.
- *Range weighted mean power:* The amplitude of electromagnetic wave reflected from metal surfaces, as in case of vehicles, is expected to be higher than that of weak reflecting surfaces like pedestrian body. This feature is therefore used for representing the reflection characteristic of an object. However, the reflection power of objects that are in a closer range to the sensor is greater than that of the object at a higher range. In order to consider this range dependency, a weight is given to the power

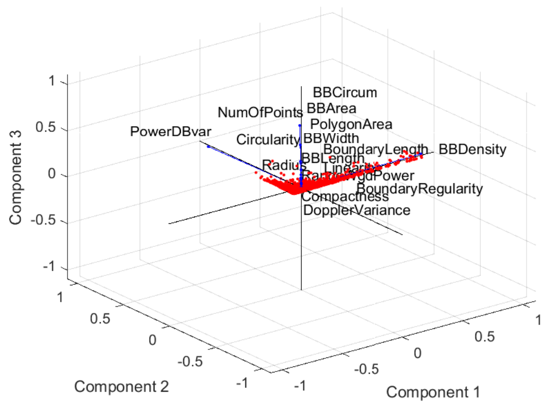


Fig. 4: Biplot of all the features projected onto first 3 PCs

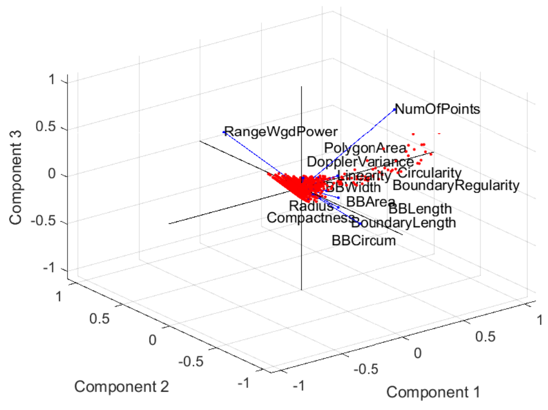


Fig. 5: Biplot of reduced features projected onto PCs

of a reflection point in the cluster. The weight should be inversely proportional to the range

$$RangeWgdPower = \frac{\sum_j w_j A(j)}{n} \\ \text{where } w_j = \frac{1}{r_j} \tag{12}$$

- *Power variance:* This feature represents the variance of reflection power in a cluster and is denoted by $PowerDBvar$. Objects with regular reflecting surfaces like passenger car is expected to have a lesser variance of the reflected power through out it's contour, whereas a bicycle for example can have a higher variance.

1) Dimensionality reduction: The relevance of all the features described above for class separation is usually not known explicitly. PCA is a widely used method in machine learning field, to analyse the contribution of each feature to the classification problem and subsequently for dimension reduction. The feature vectors are projected onto a lower dimension with the help of the principal components (PC). The so called PCs are the vectors that capture the highest variance of the data. Figure 4 shows the bi-plot with corresponding data projected onto the first three PCs and also the coefficients representing the variance contribution. As can be seen, the features $BBDensity$ and $PowerDBvar$ dominates the first two PCs and other features are entangled in the subsequent higher dimensional PCs. The feature number of points $NumOfPoints$ and the features representing dimensions, area and density are highly correlated, pointing in similar direction in the bi-plot. A bi-plot illustrating the contribution of these features, neglecting the $BBDensity$ and $PowerDBvar$ is given in Figure 5.

2) Remark: Many of the features like $NumOfPoints$ are dependent on the actual position of the object with respect to sensor. Consequently many features like circularity, radius, bounding box fit, area etc. are dependent on the $NumOfPoints$ in the cluster. But at a higher distance from sensor, as even a passenger car may only show same characteristic features as a pedestrian, considering these features in such conditions would cause a bias in class separability. On solution to overcome this would be to divide the overall classification into two sub-methods, considering the position dependability of the features. As the radar can measure the Doppler velocity of the object, it could be used as a fallback feature for the cases where the main classifier cannot give class probabilities based on all the defined features. But in conditions where the object moves with the same velocity as the ego-vehicle, this would again cause a higher misclassification. In most of the autonomous driving use cases, the detections of the sensors are fed into tracking algorithms to build a stable object state. Therefore for the condition with $NumOfPoints < 3$, the classification could be purely based on the estimated velocity of the tracked object. Moreover, the classification output could actually be a fused result of the track velocity and the result of the classifier, using some evidence based fusion techniques such as Dempster-Shafer method.

IV. Classification

Pedestrians and bicyclists are obviously very prevalent urban environments, than in highways. Also most of the automotive grade short range radar sensors with high resolution are used for near range sensing applications such as turning assist or assisted parking. In such cases, object is expected to be in the sensor FoV for only a relatively shorter time interval. Which from the other point of view means that in order to have many training samples, many hours of test data needs to be logged for training the classifier. Support Vector Machine (SVM) is one of the classification methods that is least affected by the training data sample size [13]. Therefore for object classification, SVM is used in this work. As mentioned in the previous section, the validity of feature extraction depends highly on the position of the object in the sensor's FoV. The azimuth separation for the classifier is empirically selected at $-60°$and $+60°$, based on many measurement observations.

A. SVM for classifying road users

Since it's introduction in [14], SVM has been widely used in vehicle environment perception applications like image based detection of objects, vehicle maneuver prediction etc. SVM is a maximum margin classifier, which is able to derive a decision boundary between various classes in question. Basically the boundary is an hyperplane, calculated by mapping the inputs to higher dimensional feature space. More formally the problem can be stated as, given a set of training data $\{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$ with $\mathbf{x}_i \in \mathcal{X}$ as input data and $y_i \in \mathbb{C} = \{-1, 1\}$ the class labels, the SVM calculates a decision function $F(x)$, which can then be used for predicting the value y_k for an input $\mathbf{x_k}$ at the time k [15]. If the data in $\mathbf{x}$ are linearly separable, the decision boundary separating the classes would be the optimal hyperplane in the input space $\mathcal{X}$, which is given by the decision function

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x}_i) + b = 0 \tag{13}$$

where the parameters $\mathbf{w}$ and b are the parameters to be determined by an optimisation problem. The distance between the origin and the optimal hyperplane is then given by $\frac{b}{\|\mathbf{w}\|_2}$ with $\|\mathbf{w}\|_2$ representing the least-squares norm of $\mathbf{w}$. As the hyperplane is the boundary that separates the two classes, the condition satisfying each of the class can be written as

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b &\geq +1, \text{ for } y_i = +1 \\ \mathbf{w}^\top \mathbf{x}_i + b &\leq -1, \text{ for } y_i = -1 \end{aligned} \tag{14}$$

which can be combinedly written as an inequality

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \tag{15}$$

Intuitively, for distinction of the classes, the margin between two classes separated by the optimal hyperplane need to be maximized. The distance between the optimal hyperplane and $\mathbf{x}$ is $\frac{y_i(\mathbf{w}^\top \mathbf{x}+b)}{\|\mathbf{w}\|_2}$. Therefore the margin distance between the two classes from (14) are

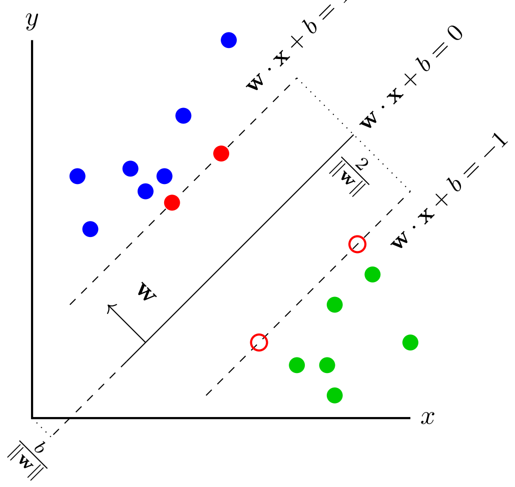
$$M = \frac{1}{\|\mathbf{w}\|_2} - \frac{-1}{\|\mathbf{w}\|_2} \Rightarrow \frac{2}{\|\mathbf{w}\|_2} \tag{16}$$

The optimisation problem for linear case is then to maximize the margin M or in other words to minimize $\|\mathbf{w}\|_2^2$, which is formulated as [15]

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|_2^2 \\ &\text{subject to } \ y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \\ &\qquad i = 1, 2, \ldots N \end{aligned} \tag{17}$$

The above problem is called the *hard-margin* SVM. However, in many cases there is no direct linear separability of the classes. In that case the problem is solved by including a slack variable ξ_i representing the margin failure and regularisation constant C

$$\begin{aligned} &\min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{subject to } \ y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\ &\qquad \xi_i \geq 0, i = 1, 2, \ldots N \end{aligned} \tag{18}$$



y
x
$\mathbf{w} \cdot \mathbf{x} + b = 1$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
$\mathbf{w} \cdot \mathbf{x} + b = -1$
$\frac{2}{\|\mathbf{w}\|}$
$\frac{b}{\|\mathbf{w}\|}$
$\mathbf{w}$

Fig. 6: Linear SVM

and is called as *soft-margin* SVM. Moreover, when the classes are not linearly separable in the input space $\mathcal{X}$, they can however be mapped using to a higher dimensional feature space $\mathcal{F}$ by the mapping $\Phi : \mathcal{X} \mapsto \mathcal{F}$. The classes are then again linearly separable in the higher dimensional feature space $\mathcal{F}$ as illustrated in Figure 7.

In [14], Lagrangian method to solve the above optimisation problem (18) in the dual space is derived. With Lagrangian multipliers α_i and β_i, the Lagrange functional is given as

$$L = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\mathbf{w}^\top \ell(\mathbf{x}_i)) + b) - \sum_i \beta_i \xi_i \tag{19}$$

and hyperplane is the solution of the optimisation problem in it's dual form

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x_i})^\top \phi(\mathbf{x_j}) \\ &\text{subject to } \sum_{i=1} N\alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C \end{aligned} \tag{20}$$

The mapping and the dot product computation is realised using a kernel function $K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^\top \phi(\mathbf{x_j})$. There are many kernel functions that can be used for SVM as detailed in [16]. Radial Basis Function (RBF) kernel is used in this work. Therefore, the classification decision function is given as

$$F(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i e^{-\gamma\|x_i - x\|^2} + b\right) \tag{21}$$

B. *Implementation and parametrisation of the SVM*

So for the implementation of the above formulated SVM problem, two parameters need to found: C which controls the softness of the margin and γ which controls the hyperplane curvature for non-linearity. For the implementation of the SVM, libsvm library of [17] is used. In order to build the classifier, the SVM needs to be trained first. The entire

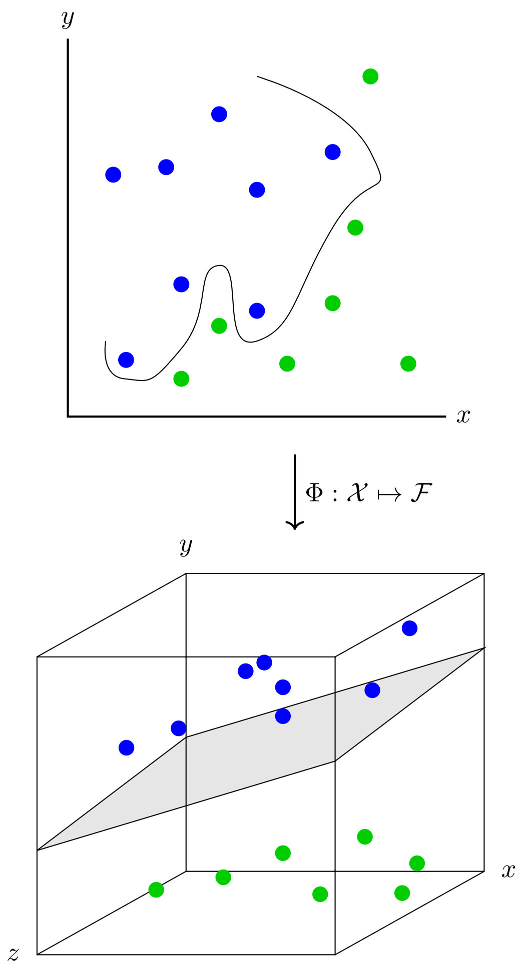


Fig. 7: Non-linear SVM

feature data with 2658 samples is scaled to the range $[0, 1]$ as recommended by the authors of libsvm. 66.6% of the data set is divided as training data and the remaining 33.3% as testing data. The multi-class classification problem is solved as one-vs-rest classification problem in libsvm. Before the actual training of classifier, the parameters need to be estimated by cross-validation method. Cross-validation is the method of dividing the training set into certain fold of subsets, train the classifier with a set of subsets and sequentially test the classifier with one subset which was not used in training.

Grid-search on C and γ using cross-validation is recommended in [17] and used so in this work. Sequences of (C, γ) are tried and the one with the best cross-validation accuracy is selected. Initially a rough grid-search with 3-fold cross-validation and step-size 2 is made with broad (C, γ) ranges $C = 2^{-20}, 2^{-18}, \ldots, 2^{20}$ and $\gamma = 2^{-20}, 2^{-18}, \ldots, 2^{20}$. This is to roughly find the parameter regions where the cross-validation accuracy is high. As depicted in Figure 8, the best pair obtained is $(2^{18}, 2^{2})$ with cross-validation accuracy of 84.3115% and the regions bounded by $C = 2^{-4}, \ldots, 2^{20}$ and $\gamma = 2^{-12}, \ldots, 2^{12}$ has higher cross-validation accuracy. However, the number of support vectors (SV) increases by

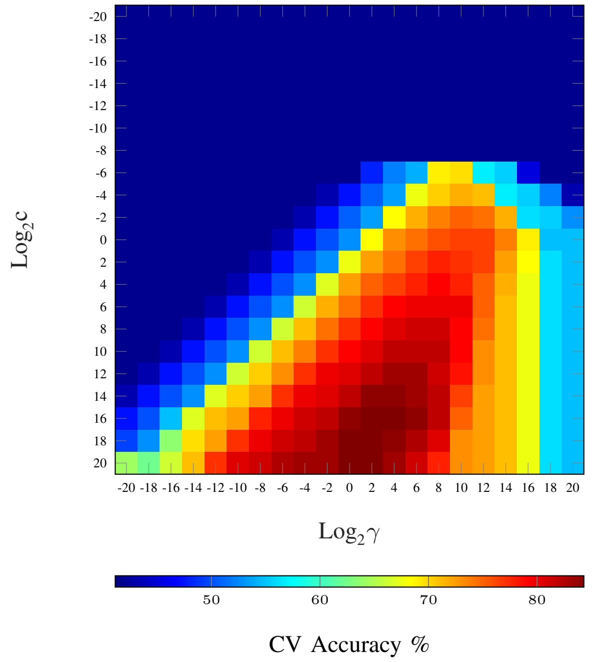


Fig. 8: Rough RBF-Kernel parameter search by cross-validation with step-size 2

increasing the slack value i.e. C. This consequently increases the prediction complexity because for every test point, a dot product with each SV needs to be computed. On the other hand, a very low value of C would increase misclassification. A higher value for γ would result in over-fitting and a very low value would not reflect the non-linearity in the decision boundary. Therefore a trade-off is to be made between the cross-validation accuracy and kernel parameters. So in the second medium-scale grid-search the scale of (C, γ) is reduced to the range $C = 2^{-2}, 2^{-1}, \ldots, 2^{18}$ and $\gamma = 2^{-6}, 2^{-5}, \ldots, 2^{8}$ with a step-size 1. The best pair from medium scale search is $(2^{16}, 2^{3})$, with cross-validation accuracy of 84.0293%, as shown in Figure 9. Finally, the search is narrowed down to the region $C = 2^{4}, 2^{3.75}, \ldots, 2^{13}$ and $\gamma = 2^{-2}, 2^{-1.75}, \ldots, 2^{6}$ with a step-size 0.25, as depicted in Figure 10 and the best parameter is found to be $(2^{13}, 2^{3.5})$ with a cross-validation accuracy of 82.9007%. Considering the trade-off, the parameter pair $(2^{10.5}, 2^{4.25})$ providing a cross-validation accuracy of 81% is chosen for training the model, along the direction of best parameter in the grid.

V. Experiments and Results

A. Properties

A data set consisting of detections from a prototype radar sensor with a carrier frequency of the range 77-81 GHz is used. The range and Doppler of a target are measured by FMCW principle. The used radar has an opening angle of $\pm 75^{\circ}$, angular resolution of $1^{\circ}$and range resolution of 0.1m. The radar sensor delivers measurements in polar coordinates with azimuth, range and relative velocity values of objects. The FoV covered by the radar sensor as mounted on the right

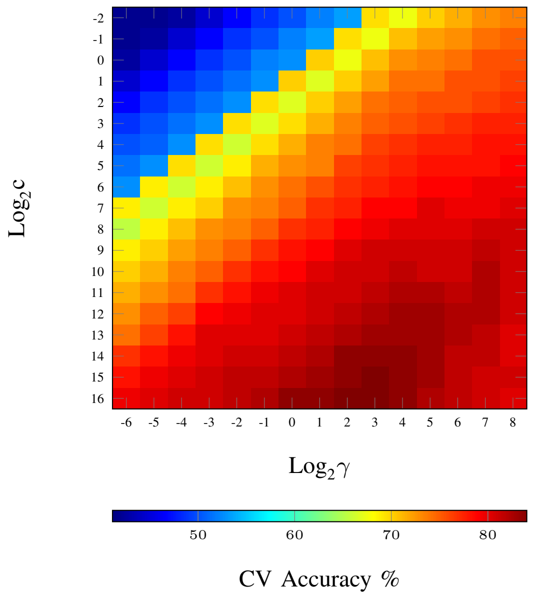


Fig. 9: Medium scale RBF-Kernel parameter search by cross-validation with step-size 1

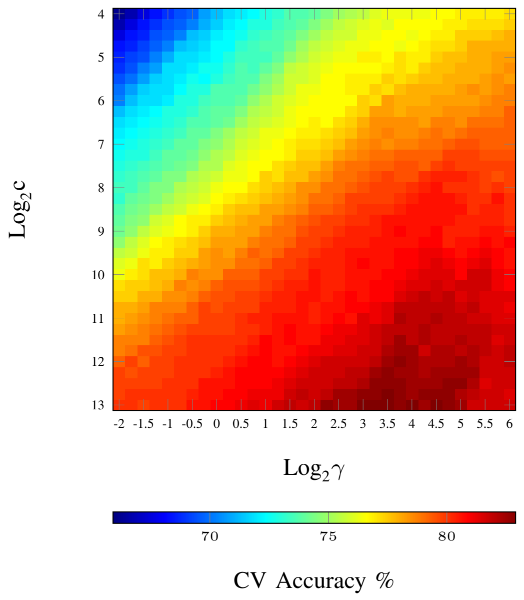


Fig. 10: Small scale RBF-Kernel parameter search by cross-validation with step-size 0.25.

corner of the vehicle is illustrated in Figure 11. Further, the object classes of interest are pedestrian, bicyclist and passenger car. Many sets of measurement are made for each object class, with different maneuvers, in order to capture a wide spread of feature data. A bigger test area of radius more than 60 m is chosen in order to avoid the reflections that could come from other objects in the test area, which would cause a bias in feature extraction. However, sensor clutter cannot be avoided explicitly but are removed from measurements similar to the preprocessing method explained in section II. After these

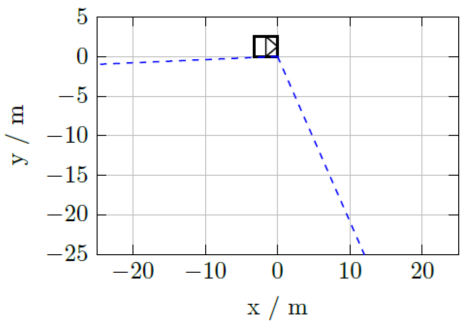


Fig. 11: Radar sensor field of view

pre-processing steps, a total of 2658 labelled cluster samples are considered further for feature extraction and also for the classifier training.

B. Clustering results

Considering the properties of the used radar sensor, the parameters *Eps* and *MinPts* required for DBSCAN algorithm are chosen as 3 and 0.05 m respectively. A snapshot of the scenario for testing detections clustering and the corresponding result is shown in Figure 12. In the shown test scenario, a bicyclist moves parallel to the ego-vehicle. The sequence is recorded in a yard, where a large wall and metal doors in the background, which are also detected and grouped as distinct clusters. Although the radar delivers only lesser detection

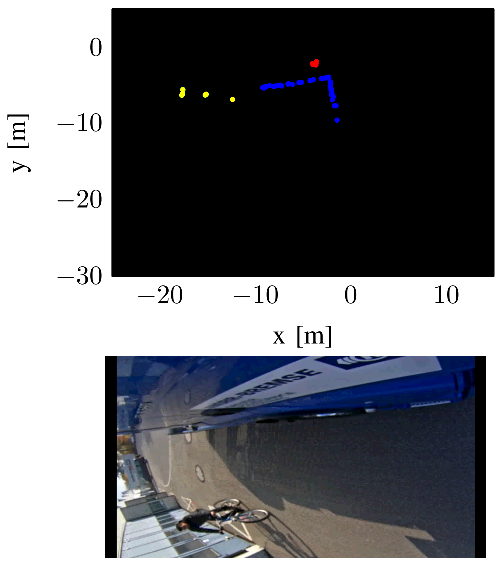


Fig. 12: Snapshot of a test sequence showing the results of detections clustering with DBSCAN. In the top Figure the cluster detections belonging to bicyclist are indicated by red markers, cluster detections belonging to corner of building are indicated in blue markers and unclustered individual detections in yellow. Bottom Figure is the corresponding web-cam snapshot

points compared to other high resolution sensors like laser

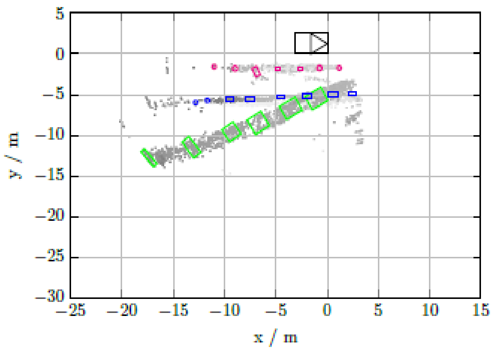


Fig. 13: Example measurement scenario illustrating geometric shape fit.

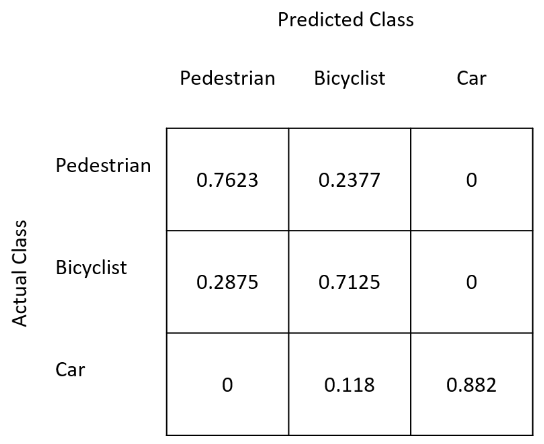
scanner, the results show that the clustering is still feasible with effective heuristics. Distinction of objects when they are too close to each other is possible only to a certain extent using a single radar sensor. Also, when a poor reflecting class like pedestrian is very close to other object class with high reflectivity like a passenger car, it may happen that the interference is high and the pedestrian may not be detected at all. In addition to the radar sensor, using multiple sensors like camera or laser scanner and fusion of detections from these multiple sensors would improve the object distinction.

C. Feature extraction results

Another test for evaluating the geometric shape and feature extraction was performed, this time also involving the pedestrian and passenger car apart from the bicyclist. The shape fitting result is presented in Figure 13. The green bounding boxes belong to the passenger car, blue to the bicyclist and magenta to the pedestrian. For clarity purpose, the shape fit of pedestrian and bicyclist are shown only for their forward motion.

D. Classification results

The confusion matrix of the SVM classifier for the pedestrian, bicyclist and car classes are depicted in Figure 14. The rows of the confusion matrix represent the true class of the object and the columns represent the predicted class. The diagonal elements of the confusion matrix represent the true-positives (TP) of that specific class and the corresponding row elements represent the false-negatives (FN). The car class has the highest probability of TPs, where it is classified correctly for 88.2% of the time. The confusion of the car class with pedestrian class is 0 %, whereas there is a small confusion of 11.8 % with the bicyclist class because the bicyclist samples include Doppler measurements overlapping with the car class. The pedestrian class is classified correctly 76.23 % of the time and the bicyclist is classified correctly 71.25 % of the time. There is relatively larger confusion between the pedestrian and the bicyclist class because many of the features from radar detections are identical to both of the classes, although the Doppler difference is wider. As both



Actual Class		Predicted Class		
		Pedestrian	Bicyclist	Car
	Pedestrian	0.7623	0.2377	0
	Bicyclist	0.2875	0.7125	0
	Car	0	0.118	0.882

Fig. 14: Confusion Matrix representing the results of radar based object classification. Numbers represent the probability values.

pedestrians and bicyclists are relevant objects for most of the accident avoidance applications like an urban turning assist, it is important to reduce the false alarms due to irrelevant objects like a passenger car. This requirement can very well be satisfied by the developed classifier.

VI. Conclusion

This paper presented an approach to classify road users based on high-resolution radar sensors for autonomous driving applications. The suggested procedure begins with preprocessing the radar detections and outputs the class of the objects in the vehicle environment. After an initial preprocessing, the radar detections were clustered into distinct groups using DBSCAN clustering algorithm. Apart from the geometric features, different radar specific characteristic features were defined and consequently extracted for each of the detection clusters. A SVM classifier was constructed and parametrised, which takes the extracted features as an input and gives the class probability of the object as output. The clustering, feature extraction and classification performance were demonstrated based on different evaluations. The classifier shows promising results for separating passenger cars, bicyclists and pedestrians. However, there is still high confusion between the pedestrian and bicyclist classes. This can be improved in the future by using different classification approaches such as deep neural networks and also by using a bigger dataset recorded with mutiple radar sensors.

References

[1] D. Stüker, Heterogene Sensordatenfusion zur robusten Objektverfolgung im automobilen Straßenverkehr. PhD dissertation, Carl von Ossietzky-UniversityOldenburg, 2006.
[2] M. Munz, Generisches Sensorfusionsframework zur gleichzeitigen Zustands-und Existenzschätzung für die Fahrzeugumfelderkennung. PhD dissertation,Ulm University, 2011.
[3] E. Schubert, M. Kunert, A. Frischen and W.A. Menzel, Multi-Reflection-Point Target Model for Classification of Pedestrians by Automotive Radar. In:Proceedings of the 11th European Radar Conference, pages 181–184, 2014.

[4] F. Fölster, Erfassung ausgedehnter Objekte durch ein Automobil-Radar. PhD Dissertation, Technical University Hamburg-Harburg, 2006.

[5] N. Kämpchen, Feature-level fusion of laser scanner and video data for advanced driver assistance systems. PhD dissertation, Ulm University, 2007.

[6] H. Zhao, X.W. Shao, K. Katabira and R. Shibasaki, Joint tracking and-classification of moving objects at intersection using a single-row laser range scanner, in:Proceedings of the IEEE Intelligent Transportation Systems Con-ference, pages 287–294, 2006.

[7] C. Premebida, O. Ludwig, and U. Nunes, Exploiting LIDAR-based Features on Pedestrian Detection in Urban Scenarios. In:Proceedings of the 12th Inter-national IEEE Conference on Intelligent Transportation Systems, pages 18–23, 2009.

[8] M. Ester, H.P. Kriegel, J. Sander and X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. in:Proceedingsof KDD, vol. 96, pages 226–231, 1996.

[9] V. Nguyen, A. Martinelli, N. Tomatis and R. Siegwart. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In: IEEE/RSJ Conference on Intelligent Robotics and Systems, 2005.

[10] K. O. Arras, O. M. Mozos and W. Burgard. Using Boosted Features for the Detection of People in 2D Range Data. In: IEEE International Conference on Robotics and Automation, pages 3402–3407, 2007.

[11] C. Premebida, O. Ludwig and U. Nunes. Exploiting LIDAR-based Featureson Pedestrian Detection in Urban Scenarios. In:Proceedings of the 12th Inter-national IEEE Conference on Intelligent Transportation Systems, pages 18–23,2009.

[12] S. Pietzsch. Modellgestützte Sensordatenfusion von Laserscanner und Radar zur Erfassung komplexer Fahrzeugumgebungen. PhD dissertation, TechnicalUniveristy Munich, 2015.

[13] C. Li, J. Wang, L. Wang, L. Hu and P. Gong. Comparison of Classification Algorithms and Training Sample Sizes in Urban Land Classification with Landsat Thematic Mapper Imagery. In: Remote Sensing, vol. 6, no. 2, pages 964–983, 2014.

[14] C. Cortes and V. Vapnik. Support-Vector Networks. In: Machine Learning, pages 273–297, 1995.

[15] B. E. Boser, I. M. Guyon, V. N. and Vapnik. A Training Algorithm for Optimal Margin Classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[16] M. Hoffmann, Support Vector Machines — Kernels and the Kernel Trick, 2006.

[17] C.-C. Chang and C.-J. Lin, LIBSVM: a Library for Support Vector Machines, Initial version: 2001 & Updated version 2013. https://www.csie.ntu.edu.tw/ cjlin/libsvm/.