Noname manuscript No.
(will be inserted by the editor)

Track-based self-supervised classification of dynamic obstacles

Roman Katz · Juan Nieto · Eduardo Nebot · Bertrand Douillard

the date of receipt and acceptance should be inserted later

Abstract This work introduces a self-supervised architecture for robust classification of moving obstacles in urban environments. Our approach presents a hierarchical scheme that relies on the stability of a subset of features given by a sensor to perform an initial robust classification based on unsupervised techniques. The obtained results are used as labels to train a set of supervised classifiers. The outcomes obtained with the second sensor can be used for higher level tasks such as segmentation or to refine the within-clusters discrimination. The proposed architecture is evaluated for a particular realization based on range and visual information which produces track-based labeling that is then employed to train supervised modules that perform instantaneous classification. Experiments show that the system is able to achieve 95% classification accuracy and to maintain the performance through on-line retraining when working conditions change.

Keywords Self-supervised learning · Obstacle classification · Learning and adaptive systems · Intelligent vehicles

1 Introduction

Accurate classification of dynamic obstacles from a moving vehicle is a vital component in any architecture developed to achieve some kind of autonomy (see, for example, the urban Darpa Grand challenge DARPA (2007)), or to provide situation awareness information to drivers (Sun, Bebis, and Miller 2006). In either case, the considered classes usually determine different responses or levels of assessment related to the situation. Class information can be integrated, for instance, within the global navigation architecture, in obstacle avoidance, mapping or tracking modules. These classes can be used to trigger/manage the corresponding alarms or actions in driver assistance systems for commercial cars.

Perception and classification of moving obstacles in urban environments is a particularly challenging task. Several different reasons contribute to this. First of all, there might be many different obstacles (with particular dynamics) involved in this scenario, such as pedestrians, cars, trucks, and bikes. The observer should be able to first detect these various agents from different positions/angles and cope with occlusion, while considering at the same time its own dynamics. Environmental conditions could affect the performance not only in terms of perception, but also in the classification. Even when a classifier might be well posed for some particular scenario (for the one it has been trained), changing conditions might indeed affect its accuracy while acting beyond the range where it can generalize.

In this work, we present a multi-sensor architecture to perform automatic moving obstacle classification. As one of the principles of this work, we perform fusion of a sensor particularly well suited to perform a task, together with others that complement the architecture and provide robustness. The second core idea in this work is the necessity to have an unsupervised module in our architecture. This is not only motivated by the constraints linked to hand labeling in supervised techniques, but also by the need of a classifier which adapts internal models when the working conditions change beyond its generalization capabilities.

We propose a hierarchical architecture that relies on the stability of a subset of features given by a sensor to perform an initial robust classification based on unsupervised techniques. The obtained results are then used as labels to train a set of supervised classifiers. Although the architec-

Roman Katz · Juan Nieto · Eduardo Nebot · Bertrand Douillard
ARC Centre of Excellence for Autonomous Systems
Australian Centre for Field Robotics
The University of Sydney
Sydney, NSW 2006, Australia
E-mail: r.katz@acfr.usyd.edu.au

ture is general and can be instantiated in a variety of ways, we demonstrate the applicability and validity of the concepts for a particular instance using range and visual information. A track-based labeling is first achieved using laser data, which is then employed to train a supervised modules to perform instantaneous classification. We introduce a multi-resolution grid representation for the tracks captured by the laser named *stamp*. By using stamps, shape and dynamic information can be robustly processed, and accurate labels obtained in an unsupervised manner. The generated labels train a set of supervised classifiers, based on the laser itself and on visual information from a monocular color camera.

The structure of this paper is as follows. We first present related work in Section 2. Our general self-supervised approach is introduced in Section 3. Key concepts and main contributions are described, with emphasis on a particular instance based on laser and visual information. Section 4 explains how an accurate unsupervised labeling can be obtained by using track information of dynamic obstacles by means of stamps. Section 5 presents the use of these unsupervised labels for training supervised classifiers. Section 6 illustrates the performance of the system through experiments, focusing on (i) generalization and (ii) adaptive capabilities. Generalization is explored by evaluating the classification performance of the system using all the incoming, instantaneous sensory information. Adaptability is evaluated by applying the self-supervised architecture to online retraining. Conclusions and future work are finally discussed in Section 7.

2 Related work

There has been extensive research on object classification in machine learning and robotics, in terms of supervised and unsupervised classifiers, and their different possible combinations. Supervised approaches are usually more accurate than unsupervised ones because the training data contain the input vectors, together with the corresponding labels. Unsupervised schemes, on the other hand, do not use labeling information and must act only based on the input vectors, usually clustering groups within the data (Duda et al 2001). The integration of these two paradigms might have different goals. When considering partially labeled datasets (sets with labeled and unlabeled samples), the combination can be performed to bias a supervised classifier using information coming from the unsupervised module. *Semi-supervised* learning (Cohen et al 2002), for instance, considers large amounts of unlabeled data, and a small subset of labeled one. This has, in fact, important implications since it is often very expensive to collect labeled data in real applications. Manual labeling can be actually infeasible in autonomous robotics. Our approach falls in the category of unsupervised techniques with automatic generation of labels. This automatic label generation becomes a core concept within an architecture aimed at performing robust classification, in long term navigation with changing environments.

Self-supervised schemes using no labels have also been proposed in the literature. Weber et al (2000) propose visual object classes that are automatically learned, by first identifying salient features and then performing robust clustering. The work presented by Luo and Savakis (2001) approaches visual segmentation as a hierarchical, two-stage classification process. The first stage performs unsupervised clustering on the images, identifying regions of high confidence. This information is later used to train a supervised classifier that acts on the low-confidence regions. These papers relate to our approach in the sense that salient, stable features are also used in a first, unsupervised procedure. In our case, rather than identifying high confidence regions sequentially, this invariance is analyzed a priori in the features' space, such that a posterior unsupervised clustering can achieve high-confidence labeling.

Some other approaches perform unsupervised classification of dynamic objects using spatio-temporal representations. The work by Luber et al (2008) performs classification by obtaining exemplar-models for representing varying appearance of objects in 2D laser scans. Their exemplar representation, as opposed to the one by Schultz (2006), integrates both classification and tracking, and is able to learn various models (such as pedestrians, skaters) autonomously. Our work is closely related to theirs, in the sense that we also aim at capturing time-varying appearance of dynamic obstacles for classification. However, we use in our work a different representation for fusing range information in general urban environments, a scenario that tends to present very different challenges in terms of planarity, complexity and involved dynamics.

Several papers have presented self-supervised classification architectures in robotics applications, mostly focused on terrain classification. Here, "supervisors" or labeling modules that are based on one kind of sensor usually train separate classification modules with different sensory inputs. The work by Brooks and Iagnemma (2007), for instance, presents a self-supervisory mechanism that learns visual appearance of terrain classes based on vibration sensing. The acquired knowledge is then transferred to a visual classifier that identifies terrain classes in the distance. A similar approach is taken by Stavens and Thrun (2006), where an inertial sensor based estimator learns terrain roughness in an unsupervised fashion, then the data trains a laser to detect rough terrain. The work by Dahlkamp et al (2006) identifies traversable terrain by combining long range monocular camera with short range laser scanner. This paper addresses

self-supervision in autonomous navigation, with particular emphasis on the classification of dynamic obstacles.

3 Overview and contributions

This section introduces our approach for dynamic obstacle classification. Key concepts of the self-supervised architecture are first described, with emphasis on an instance based on 2D laser and visual information. Although the proposed scheme is general and can be applied to many different situations, we aim at dealing with this popular scenario in sensor-based autonomous navigation. The main contributions of this work are then summarized.

3.1 Overview

This paper proposes a self-supervised architecture for robust classification of moving obstacles in urban environments. A high-level diagram with the main functional components of our scheme is shown in Figure 1. Two fundamental types of processing are presented in the proposed scheme; track-based classification and instantaneous classification.

The first stage of the architecture considers *track-based classification* of dynamic obstacles. The "batch" information contained in the tracks is here used to produce accurate labeling that is then employed to train supervised modules that perform instantaneous classification. The goal here is to use unsupervised algorithms together with pre-processed laser data in order to produce an initial classification. Due to the considered track-based representation, this initial stage is not suitable for performing online classification and therefore this class information is transfered to modules that employ a different set of features to achieve *instantaneous classification*.

The reason for propagating accurate unsupervised classification of tracks to supervised modules is two-fold. First, classification can be extended by training classifiers using more general features that do not depend on particular conditions, and might use different or complementary sensory modalities. Second, adaptability capabilities can be incorporated in the system. This is indeed a very important feature regarding robust classification for long term autonomous navigation. Let's consider, for instance, a scenario where working conditions can change drastically. These changes might affect the performance of one particular sensor if no online tuning is available in the system. However, if the system can adjust (or retrain) regularly based on updated conditions, then it could maintain a robust performance over time.

3.2 Contributions

The main contributions of this work are: a procedure for accurate unsupervised generation of labels using a new tracks representation based on *stamps*, and its integration within a self-supervised learning architecture.

- The first contribution is a methodology for performing accurate labeling using tracks, depicted as the top pipeline in Figure 1. The scheme first identifies persistent tracks using a hidden Markov model (HMM). The approach uses a novel tracks representation using stamps, which gives a robust basis for shape synthesis. Stamp features are finally extracted and discriminant ones selected for accurate unsupervised label generation by clustering.
- The second contribution is the integration of the previous methodology for unsupervised generation of labeling within a self-supervised learning architecture. The transfer of information from the inferred unsupervised classes to supervised modules (as illustrated in the bottom N processes in Figure 1) provides the system with the ability to better generalize to different sensory inputs and features. Moreover, online long term adaptability capabilities are included in the classification architecture.

4 Temporal-based classification: unsupervised generation of labels

This section presents how accurate labeling can be performed using the information in the laser tracks defined by dynamic objects. Considering a laser scanner as the base sensor, we can extract laser tracks by performing detection and temporal association of moving obstacles. An approach for unsupervised generation of labels is proposed that relies on identifying persistent tracks and modeling them using stamps. A *stamp* in this work (as detailed in Sec. 4.2) refers to a range-based multi-resolution map that represents a track and provides a mechanism for data integration and feature extraction. The proposed procedure can be described through its main modules:

1. Computation of track persistency: the reliability of the stamp representation for the tracks relies on identifying those laser tracks that actually correspond to dynamic objects. The belief that a track corresponds to a moving obstacle is defined using a persistency measure, and a probabilistic approach is proposed to compute this persistency using HMM.
2. Representing tracks via stamps: stamps are constructed from persistent tracks by aligning the corresponding scans for each of the tracks and building an occupancy grid (OG) of the registered returns.

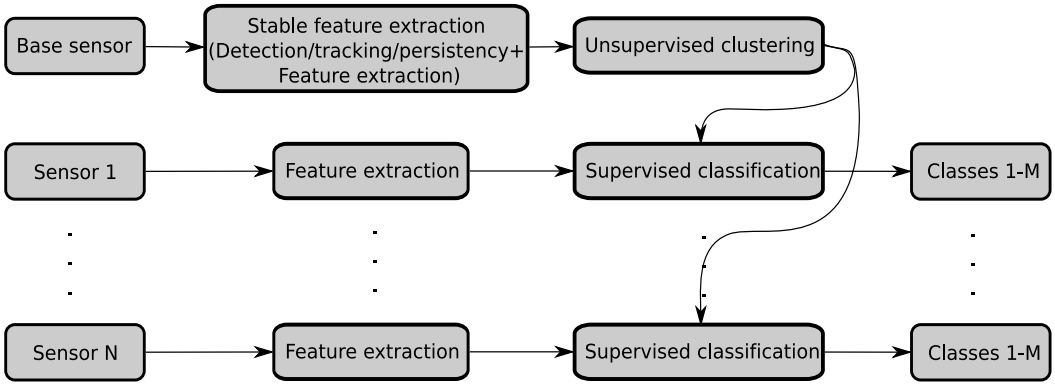


Fig. 1 High-level diagram of the self-supervised architecture. The top pipeline relies on a stable feature detection module, which takes inputs from a base sensor and feeds them into an unsupervised clustering process. The labels generated are then used to train a set of N supervised classifiers.

3. Extracting features from persistent stamps: once stamps have been obtained, features that synthesize the classes for the underlying dynamic obstacles are extracted. These features are also combined with information about the dynamics of the tracks.
4. Stamp features selection for labeling: clustering is performed on the possible subsets of features to find the optimal combination for unsupervised generation of labels.

In the following sections we describe in detail these four stages for unsupervised labeling.

4.1 Computation of track persistency

A stable feature extraction module (as shown in the top pipeline in Figure 1) needs to capture the appropriate information encoded as a vector feature representing the dynamic objects in the scene. The robustness of the subsequent unsupervised clustering procedure depends not only on the type of features, but also on the quality of the information contained in the perceived tracks. In this work, *persistency* refers to the belief that a track actually corresponds to a moving obstacle.

This section presents the use of HMMs to identify persistent tracks in a probabilistic framework. The inputs for HMM-based persistency computation are provided by the combination of tracking and motion detection modules based on laser measurements. The tracking module estimates past and present location of the dynamic objects around the vehicle, i.e., the corresponding *track* for each of the objects. The tracker uses obstacles detected as dynamic for initialization, and updates the tracks based on laser information. At the same time, sequences of HMM observations are generated for each track. These HMM observations are based on the re-detection of dynamic objects, and are obtained by combining detection and tracking. Objects that are indeed dynamic will tend to be re-detected often, and should lead to high persistency in the tracks. Conversely, false detection of static obstacles (e.g., due to noise) will not occur frequently and should decrease the persistency. Details of the HMM model for persistency computation are first introduced. The tracker and detection modules are then presented, together with their combination towards obtaining suitable HMM observation sequences. Finally, the instantiation of the observation model and transition matrix is shown, integrating observation sequences into the HMM for persistency computation.

4.1.1 HMM formulation and notation

A hidden Markov model (HMM) (Rabiner 1990) is proposed to solve the estimation of persistency for each of the tracks. Let the state X_k denote the hidden state and Y_k the observation. The corresponding graphical model for this HMM is shown in the left image of Fig. 2. In our problem, there are two possible discrete states $X_k \in \{1,2\}$, where $X_k = 1$ and $X_k = 2$ represent "persistent" and "unpersistent" states (i.e., $X_k \in \{\mathsf{Persistent}, \mathsf{Unpersistent}\}$), respectively. Effectively, X_k indicates whether the whole track is persistent given observations until time k. We find a solution to the stochastic estimation of track persistency by recursively evaluating $P(X_k \mid \mathbf{Y}_k)$ as:

$$\begin{aligned} P(X_k = j \mid \mathbf{Y}_k) &\propto P(Y_k \mid X_k = j) P(X_k = j \mid \mathbf{Y}_{k-1}) \\ &= P(Y_k \mid X_k = j) \sum_i A(i,j) P(X_{k-1} = i \mid \mathbf{Y}_{k-1}), \end{aligned} \quad (1)$$

where $A(i,j) = P(X_k = j \mid X_{k-1} = i)$ is the transition matrix expressing the probability a track changes between different states $X_k \in \{\mathsf{Persistent}, \mathsf{Unpersistent}\}$, $P(Y_k \mid X_k)$ the observation model, and $\mathbf{Y}_k$ the set of observations received until time k. The initial state distribution is given by $\pi(i) = P(X_1 = i)$. The structure of this HMM, depicting the transitions between the numbered states, is shown graphically in the right diagram of Fig. 2.

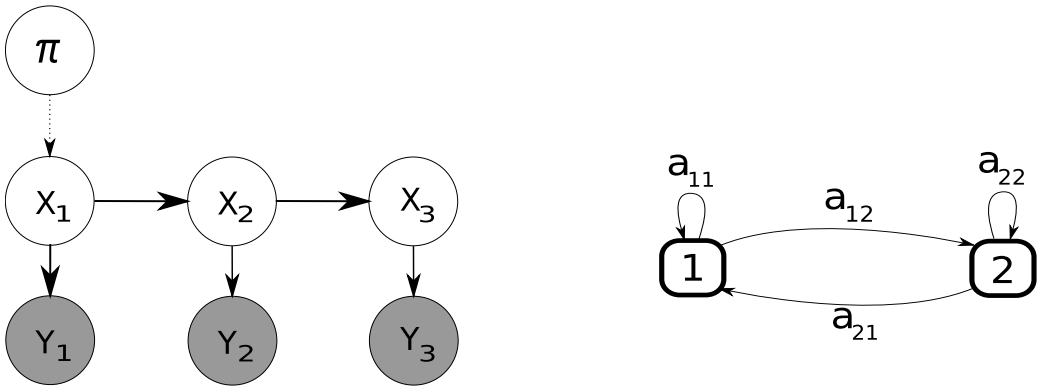


Fig. 2 HMM to compute track persistency. Left image shows the graphical model of the HMM, where X_k denotes the hidden state and Y_k its observation. The right image illustrates the HMM transition diagram, for the states $X_k \in \{1 : \mathsf{Persistent}, 2 : \mathsf{Unpersistent}\}$.

4.1.2 Observations through detection and tracking

The generation of HMM observation sequences is addressed by first integrating a motion detection module together with a multiple target tracker. The specific setup is composed as follows. A probabilistic laser based motion module based on Katz et al (2008) first detects possible dynamic behavior in the laser coordinate frame, given the current laser scan ℓ_k. This procedure can be described by the following pseudo-code function interface:

$$\left[\mathbf{z}_k^{dyn}, P_k^{dyn}\right] = \texttt{motion\_detector}(\ell_k), \tag{2}$$

where the $\mathbf{z}^{dyn}$ denotes the observation of the detected obstacles in laser coordinates, and P^{dyn} contains the corresponding segmented laser returns for the detected obstacles. The segmentation of laser returns is performed using agglomerative hierarchical clustering in a Euclidean space, applying a predefined distance threshold (in this work 0.3m).

The tracker uses the sample based joint probabilistic filter scheme by Frank et al (2003), performing multiple target tracking to estimate 2D position and orientation of obstacles $X_k^{dyn} = (x_k, y_k, \theta_k)$ at time k. The observations are provided by segments obtained from the current laser scan ℓ_k. Tracks are initialized using information provided by the motion detection module, where each track corresponds to only one dynamic obstacle in the scene. The combined use of the tracker and the detected objects allows the system to update the state and incorporate new tracks. Simultaneously, HMM observations Y_k can be generated for each track. The HMM observation

$$Y_k \in \{\mathsf{Redetected}, \mathsf{NonRedetected}\}$$

indicates whether each track is re-detected at time k (with respect to $k-1$), or it is not re-detected. This tracking procedure that maintains the state of dynamic obstacles and provides HMM observations can be denoted through the following pseudo-code function:

$$\left[Y_k, X_k^{dyn}\right] = \texttt{tracker}\left(\ell_k, \mathbf{z}_k^{dyn}, X_{k-1}^{dyn}\right). \tag{3}$$

Sequences of Y_k from (3) are stored in $\mathbf{Y}_k$ for each of the tracks over time, and used for HMM persistency computation. Segments of laser returns P^{dyn} from (2) are also stored for each of the tracks, and used to construct the stamp representation in Section 4.2.

4.1.3 HMM model instantiation

The entire observation sequence $\mathbf{Y}_k$ for each track is used to obtain the corresponding estimation of persistency through (1). This work fuses the HMM observation by proposing a suitable HMM observation model. The idea is to use a model that favors persistency if the detector observes dynamic objects that have been previously detected. Conversely, the model slowly decreases the persistency belief if the objects are not often re-detected. This filtering aims at dealing with false positives in the dynamic obstacle detection. It also provides a refreshing mechanism for the cases where dynamic objects suddenly become static. Considering discrete observations $Y_k \in \{\mathsf{Redetected}, \mathsf{NonRedetected}\}$, we can represent the observation model as a matrix $B_k(i,l) = P(Y_k = l \mid X_k = i)$:

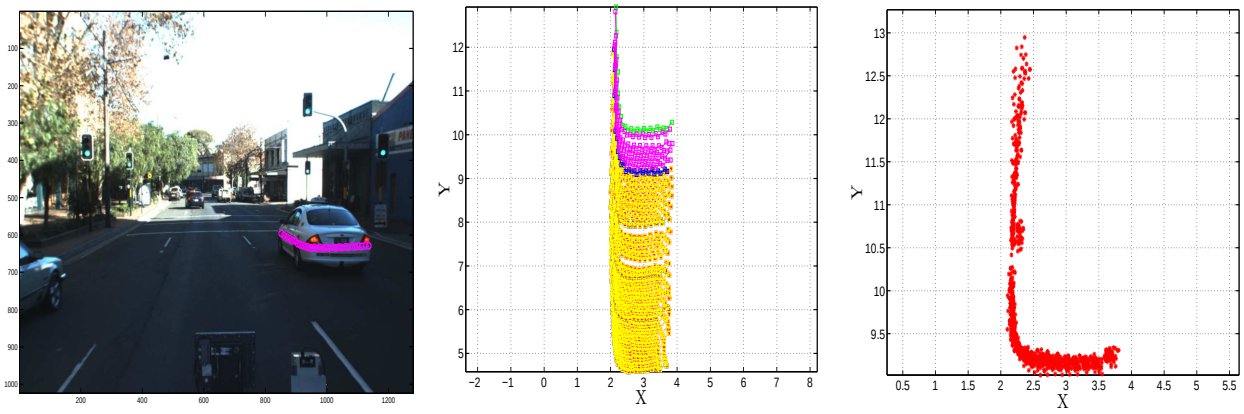
$$B_k(i,l) = \begin{bmatrix} \mathscr{C}_1 & (1-\mathscr{C}_1) \\ (1-\mathscr{C}_2) & \mathscr{C}_2 \end{bmatrix}, \tag{4}$$

with typical values of $\mathscr{C}_1 = 0.8$ and $\mathscr{C}_2 = 0.6$ empirically obtained.

The transition matrix $A(i,j)$ is learned from a manually annotated set of tracks. This is done (as in Torralba et al (2003)), by counting the number of transitions between the various states X_k. The initial state distribution is assumed uniform, that is, $\pi(i) = P(X_1 = i) = 0.5, \forall i \in \{\mathsf{Persistent}, \mathsf{Unpersistent}\}$.

4.2 Representing tracks via stamps

The aim of the method presented in this section is to obtain a robust and distinctive representation named stamp to



(a) (b) (c)

Fig. 3 Scan alignment for a track that leads to the stamp representation. (a) shows an example of car tracking, where the laser scan is projected onto the image. (b) presents the sequence of all the segmented scans extracted for the track. (c) illustrates the scans in (b) aligned using the ICP registration procedure.

describe the laser tracks. The term stamp is inspired by the modeling approach, where a distinctive mark or "signature" is used to symbolize tracks that might change over time. A *stamp* is defined as a pyramidal grid representation for a track, constructed by aligning subsequent scans that belong to the track and building a multi-level occupancy grid (Moravec and Elfes 1985) of the registered returns. This provides a solid basis for shape synthesis that deals with the difficulties for inferring spatial appearance, combining least-mean-square error alignment with occupancy grids. This occupancy grid is further extended into a multi-scale representation for increased robustness.

This section uses segments of laser returns (or just segments) P^{dyn} for each of the persistent tracks (see Section 4.1) to build the stamps. Each of the these segments contain laser returns, extracted from the original laser scans in (2). The images in Fig. 3(a)-(b) show an example of the input for generating stamps using the scans defined by persistent tracks. Considering segments of scans P^{dyn} for each of the tracks as the input, the algorithm (i) computes segment alignment, and (ii) obtains the stamps from these aligned segments. These two components are detailed below.

4.2.1 Alignment of segments

The laser segments are aligned using the iterative closest point (ICP) registration procedure by Besl and McKay (1992). ICP aligns data points which in this work are the laser returns contained in the segments. Given a set of m measurement data points $P^{dyn} = \{\mathbf{b}_i\}$ and a reference point set $X = \{\mathbf{a}_j\}$ considered fixed containing n points, ICP finds a transformation vector to align the measurement data points of P^{dyn} to the reference points of X. By recursively executed applying ICP for each of the successive segments in the track we obtain a final set $\mathbf{B}_k = \{\tilde{\mathbf{b}}_{i_1}, \ldots, \tilde{\mathbf{b}}_{i_k}\}$ of aligned scans for the track, with the index k ordering the points sequentially.

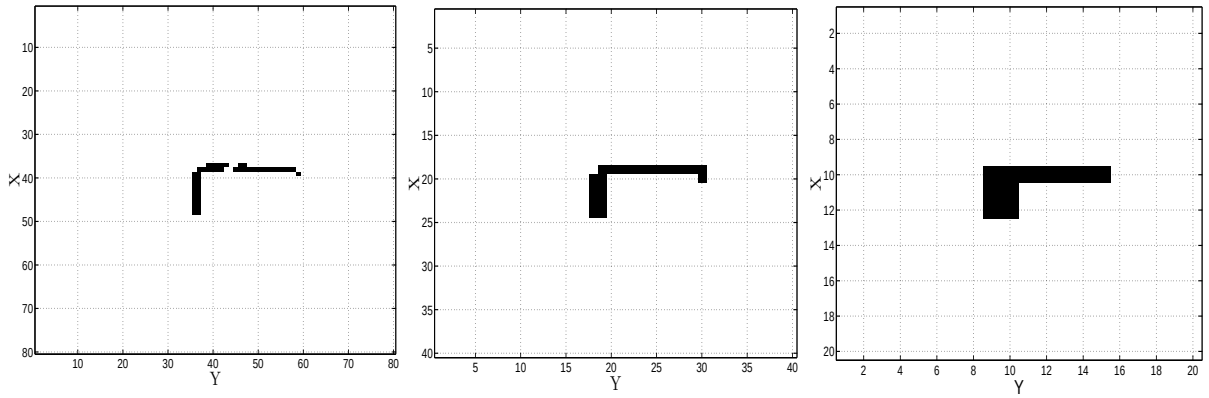
4.2.2 Stamp computation

Considering now for each of the persistent tracks the corresponding set of aligned segments $\mathbf{B}_k$, we can formulate the computation of the stamps following Elfes (1989). We first define a state variable $S(C_m)$, that stores the probability of the cell C_m being occupied $P(S(C_m) = Occ)$. Since the cell states are exclusive, $P(S(C_m) = Occ) + P(S(C_m) = Free) = 1$. The evaluation of the posterior over the occupancy of each grid cell is based on binary Bayes filters:

$$P(S(C_m) = Occ \mid \mathbf{B}_k) \propto P(\tilde{\mathbf{b}}_{i_k} \mid S(C_m) = Occ)\; P(S(C_m) = Occ \mid \mathbf{B}_{k-1})), \tag{5}$$

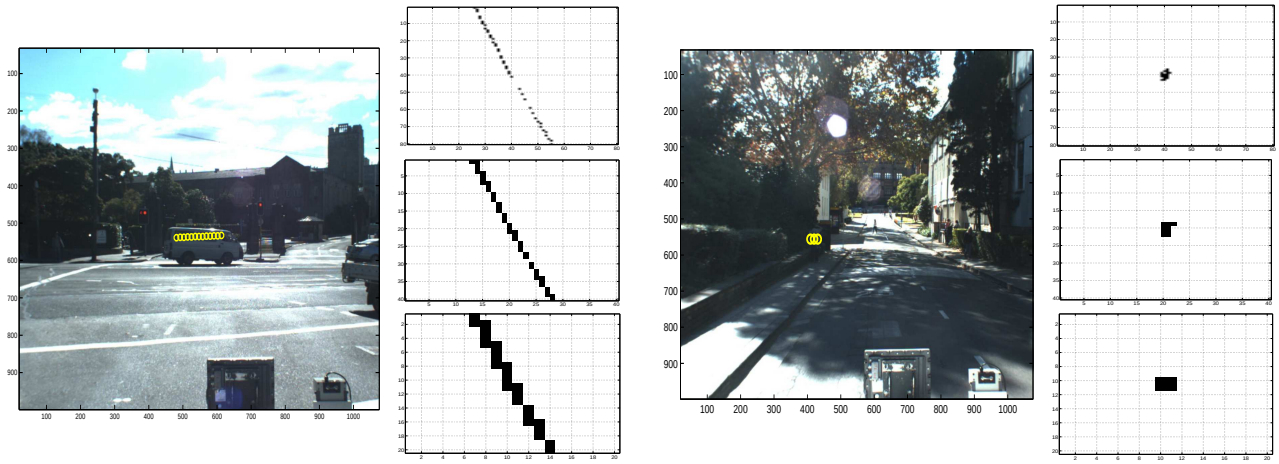
where $\mathbf{B}_k$ represents the set of observations received until time k, $P(S(C_m) = Occ \mid \mathbf{B}_{k-1})$ the previous estimate of the cell state, and $P(\tilde{\mathbf{b}}_{i_k} \mid S(C_m) = Occ)$ is the laser sensor model. A thresholding stage (with value $Thres$) is finally used to obtained a clean $Occ/Free$ representation. A full stamp can now be computed for an initial grid resolution Res_0, by computing an occupancy grid (OG) for the L grids, each with a spatial resolution that doubles the immediate finer one. A stamp of level L will then be composed of L OG layers of different resolution.

Figure 3(c) shows the final aligned laser returns for our persistent track example. Figure 4 presents the computed stamp for the example introduced in Fig. 3. The images in Fig. 4(a)-(c) show the corresponding grids for a stamp of $L = 3$ and $\mathrm{Res}_0 = 0.15m$ (for thresholding $Thres = 0.5$).



(a) (b) (c)

Fig. 4 From aligned scans to stamps. Stamp corresponding to the aligned scans for the track in Fig. 3. Images in (a),(b) and (c) show the corresponding different grids for each of the levels of a stamp of $L = 3$ and $\text{Res}_0 = 0.15m$. Note that the stamps have been rotated 90° with respect to the aligned segments in Fig. 3(c).



(a) (b)

Fig. 5 Stamps for two different track instances: dynamic car in (a), and a moving pedestrian in (b). Tracks represented using stamps with levels $L = 3$ and $\text{Res}_0 = 0.15m$. Corresponding grids for each of the stamp levels shown for both cases on the right. Note that the stamps have been rotated 90° with respect to the aligned segments.

4.3 Extracting stamp features

Once persistent stamps have been obtained, we extract features aimed at capturing the class of the dynamic object that defined the track. By using a multi-level representation for the stamps, features that can be efficiently computed on each of the levels are considered. Effectively, a compact feature scheme is used, integrating spatial features from the stamps with information about the dynamics of the tracks.

Our approach introduces the use of combined features for stamps, following the scheme presented by Tsuchiya and Fujiyoshi (2006) for visual surveillance. Specifically, stamp features include three different types of features: *shape-based* f_{Shape}, *texture-based* $f_{Texture}$, and *motion-based* f_{Motion}. The first two feature types (shape and texture-based) are computed for each $l = [1,\ldots,L]$ for a stamp of level L, whereas the motion-based features are directly derived from the tracks.

Shape-based features include $f^{(l)}_{AspRatio}$ aspect ratio and $f^{(l)}_{Compact}$ compactness. Aspect ratio measures relative size w.r.t. the size of the main axes of the stamp:

$$\text{aspect ratio} = \frac{\text{length of minor axis}}{\text{length of major axis}}. \tag{6}$$

Compactness quantifies the shape complexity of the object as a ratio between perimeter and area:

$$\text{compactness} = \frac{\text{perimeter}}{\text{area}}. \tag{7}$$

The texture-based features include $f^{(l)}_{Connect}$ 8-connectivity and $f^{(l)}_{NumCells}$ number of cells. n-connectivity is a simple measure that quantifies the smoothness/continuity of the stamp.

Table 1 Stamp features for the car (C) and the pedestrian (P) stamps from Fig. 5. Both stamps with levels $L = 3$ and $\text{Res}_0 = 0.15m$.

	$f_{AspRatio}$		$f_{Compact}$		$f_{Connect}$		$f_{NumCells}$	
	C	P	C	P	C	P	C	P
$l = 1$	0.02	0.46	7.11	3.75	0.9	4.25	41	16
$l = 2$	0.03	0.55	5.57	4	2.43	4	42	9
$l = 3$	0.06	1	5.17	5	2.83	3	23	4

It can be easily computed using masks, or by counting the n neighbors for each of the cells and adding the final count and normalizing for the complete grid. The number of cells $f^{(l)}_{NumCells}$ is a 0-connectivity texture measure which naturally computes the size of the stamp.

The images in Fig. 5 show two track instances for a dynamic car in (a), and a moving pedestrian in (b). We have represented each of these tracks using stamps with levels $L = 3$ and $\text{Res}_0 = 0.15m$, as presented in Sec. 4.2. The corresponding grids for each of the levels of the stamps are shown for both tracks on the right side of the Fig. 5(a) and (b). Shape-based ($f_{AspRatio}$ and $f_{Compact}$) and texture-based features ($f_{Connect}$ and $f_{NumCells}$) for these tracks are presented in Table 1. The values indicate the computed results for each of the features, for all the grid levels ($l = 1,2,3$) in the stamps. Columns in the table indicate features, grouping same features for both the car stamp (C) and the pedestrian stamp (P) for comparison purposes. As can be seen in the table, stamp features tend to capture the intuition of the underlying objects in most of the levels. Aspect ratio $f_{AspRatio}$ is usually small for elongated objects like cars, whereas pedestrians' ratio present more balanced overall shapes closer to unity. Pedestrians are often compact in their corresponding stamp grids, then compactness feature $f_{Compact}$ is smaller than for more irregular objects like cars. Regarding texture-based features, 8-connectivity $f_{Connect}$ tends to be small for man-made, regular objects like cars, whereas pedestrians present higher values representing more texture. Finally, the number of cells $f_{NumCells}$ captures the size of the objects accordingly for both cars and pedestrians.

The dynamics of the tracks is considered within the motion-based feature f_{Motion} for the stamps. Moving objects might present changing dynamic behavior over time. It seems sufficient, however, to just compute the maximum speed of the tracks to reason about objects' dynamics in most of the cases[1]. We therefore consider the feature $f_{Motion} = f_{MaxSpeed}$ maximum speed of the track for our stamp features representation. In order to obtain a feature vector independent of the number of levels in the stamp we first compute the minimum and maximum values for the shape and texture-based features as:

$$\begin{aligned} f_{Shape} &= \left[\min\left(f^{(l)}_{AspRatio}\right), \max\left(f^{(l)}_{AspRatio}\right), \min\left(f^{(l)}_{Compact}\right), \right. \\ &\quad \left. \max\left(f^{(l)}_{Compact}\right)\right], \\ f_{Texture} &= \left[\min\left(f^{(l)}_{Connect}\right), \max\left(f^{(l)}_{Connect}\right), \min\left(f^{(l)}_{NumCells}\right), \right. \\ &\quad \left. \max\left(f^{(l)}_{NumCells}\right)\right], \end{aligned} \tag{8}$$

where $\min\left(f^{(l)}_{(\bullet)}\right)$ and $\max\left(f^{(l)}_{(\bullet)}\right)$ are the minimum and maximum values for the feature $f_{(\bullet)}$ operating along $l = [1,\ldots,L]$ for a stamp of level L. By concatenating this shape and texture features together with the motion-based feature, we obtain the final stamp feature vector f_{Stamp} as:

$$f_{Stamp} = \left[f_{Shape}, f_{Texture}, f_{Motion}\right], \tag{9}$$

for a full stamp feature of size 9 that include four shape, four texture, and one motion feature.

4.4 Stamp feature selection for labeling

The unsupervised label generation is based on a clustering algorithm. Given a clustering algorithm, different feature spaces will result in different sets of clusters. In consequence, we evaluate all the different feature spaces by testing the clustering algorithm on all the possible subsets of features. The accuracy of the clustering for each case is evaluated using a manually labeled set[2]. It is obtained by comparing the number of matches between the cluster index and the true label.

Stamp features seem to well capture salient characteristics of dynamic objects. However, a subset of these stamp features might actually provide the best results for label generation given a particular clustering algorithm. In this section we find the optimal combination of features for unsupervised generation of labels by testing clustering on all possible subsets of stamp features. We use the *expectation-maximization* (EM) algorithm (McLachlan and Krishnan 1997; Dempster et al 1977) to find the parameters of a mixture of Gaussians that best partitions our feature space. We analyze the "optimal" subset of stamp features, that is the particular combination which provides the best separation of classes (i.e. the most discriminative features). Since for this evaluation true labels were made available through hand labeling, we compute the error for all the possible subsets.

For completeness, we compare the chosen EM clustering with some other popular algorithms. Performance is evaluated not only for EM clustering but also for hierarchical

[1] Other measures like maximum acceleration could be also considered

[2] Note that this annotated dataset is only used in the design/validation stage, similarly as the one used for the HMM. Different labeled datasets are used in the rest of the work only for accuracy evaluation.

schemes and spectral clustering. Hierarchical schemes (Duda et al 2001) perform clustering by partitioning the data using trees, based on some distance metric on the data points. Spectral clustering (Ng et al 2001) performs graph partitioning using matrices derived from the data. This is achieved by computing global similarity and partitioning the data using eigenvectors. Locally scaled spectral clustering (LS) (Zelnik-Manor and Perona 2004) has been shown to better deal with multi-scale data. In this case similarity is computed using statistics of neighboring points, such that the mean distance to the K^{th} neighbors.

These three different methods were analyzed using 153 tracks, for stamps of level $L = 3$ ($\text{Res}_0 = 0.15m$). The tracks were manually labeled for the classes 'bike', 'pedestrian', and 'car', with bike including bicycles and motorbikes, and car containing all possible automobiles (cars, trucks, etc.). Since these three classes include most dynamic objects in urban scenarios, they will be used throughout the rest of the paper. Figure 6 presents the error in clustering with respect to all possible combination of stamp features, for the evaluated clustering methods. Clustering error for the three methods is computed using all combinations of features. The error for each method is then sorted in increasing order, for a clear visualization in Figure 6. Due to this rearrangement, same error do not necessarily correspond to the same features combination in different methods.

This experimental evaluation validates our initial election of clustering approach, showing that EM achieves the best clustering performance for particular subset of stamp features. In fact, EM reaches an accuracy of 92.81% (for the error of 0.0719 shown in the plot), slightly outperforming locally scaled spectral clustering (LS) (for a local scale of $K = 5$). The optimal subsets of stamp features that correspond to the best EM clustering accuracy include these two combinations:

$$Comb_1 = \{\max(f^{(l)}_{AspRatio}), \max(f^{(l)}_{Compact}), \min(f^{(l)}_{NumCells}), f_{MaxSpeed}\}$$
$$Comb_2 = \{\max(f^{(l)}_{AspRatio}), \min(f^{(l)}_{Connect}), \min(f^{(l)}_{NumCells}), f_{MaxSpeed}\}.$$

The features in these two combinations $Comb_1$ and $Comb_2$ suggest that, in order to obtain the best clustering accuracy for unsupervised generation of labels, the stamp features aspect ratio $f_{AspRatio}$, number of cells $f_{NumCells}$, and $f_{MaxSpeed}$ must be selected. In addition, optimal subsets need to include either compactness $f_{Compact}$ or 8-connectivity $f_{Connect}$. As a matter of fact, they both quantify very similar entities, since the shapes given by compactness and connectivity features are highly correlated.

As suggested by Luber et al (2008) through the analysis of track velocities, the use of speed-based features only would not provide enough clustering accuracy for label generation. Our results are consistent with theirs: using EM algorithm and the maximum speed of the track $f_{MaxSpeed}$, we have achieved an accuracy of just 63.69%.

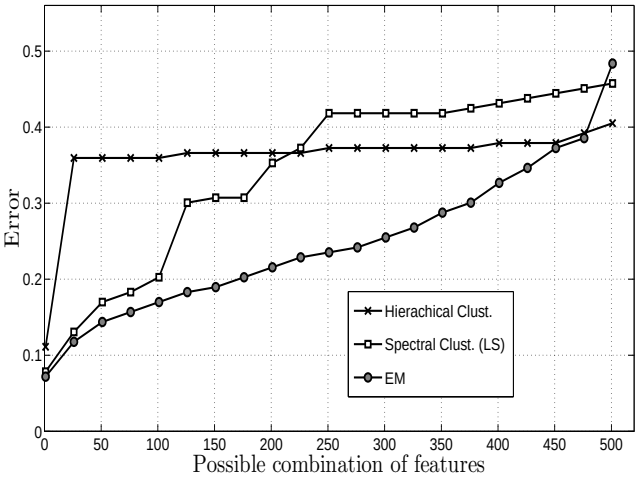


Fig. 6 Stamp features selection through clustering. Plot showing the error in clustering for different clustering methods, for all possible combination of features. EM clustering outperforms the other methods, reaching an accuracy of 92.81%.

5 Instantaneous classification

This section builds upon the architecture for self-supervised classification introduced in Section 3, focusing on the transfer of information from inferred unsupervised labels to supervised modules (bottom processes in Figure 1). The system uses unsupervised labels (computed through appropriate processing of the information provided by a base sensor) to train a set of supervised classifiers. The main objective of the self-supervised approach is to use a sensor that provides stable features, to train a noisier one. The features obtained with the second sensor can be used for higher level tasks such as segmentation or even a better discrimination within clusters. The implementations presented in this paper use laser for the unsupervised stage and cameras for the supervised one.

Having obtained unsupervised labels from laser tracks (Section 4), this section presents its utilization in training a set of supervised classifiers. Specific features based on laser and visual information are first introduced and then used in supervised classification, demonstrating the validity of the self-supervised architecture for classification. We finally combine the separate self-supervised classifiers to obtain a unique output classification result.

5.1 Features for supervised classification

We first introduce the chosen features to represent laser information that corresponds to dynamic obstacles in the scene. Features in this case are based only on geometric information and estimated velocities obtained from laser data. No filtering or additional information is included using stamp features. Here, we emphasize geometric features extracted directly from laser segments which are obtained by clustering the laser scans using a threshold in the Euclidean space of the points. Instantaneous velocities and running maximum speeds are also considered in the features. The reason for this is that, at this stage, we want to avoid the use of delayed features to be able to classify obstacles on-the-spot.

Laser features can be summarized as: f_{lsr_1} number of points in the cluster, f_{lsr_2}-f_{lsr_3} normalized point covariances in X and Y, f_{lsr_4} range to the cluster, f_{lsr_5} cross-section, f_{lsr_6} linear error (mean error of points to a fitted straight line), f_{lsr_7} 2^{nd} order error (mean error of points to a fitted 2^{nd} order curve), f_{lsr_8} curvature coefficient (2^{nd} order coefficient for the approximated curve), f_{lsr_9} L-shape error (mean error of points to a fitted L-shape open polygon), $f_{lsr_{10}}$ norm of the speed, and $f_{lsr_{11}}$ norm of running maximum speed. Considering all these individual laser features, the final laser feature vector is composed as $f_{lsr} = [f_{lsr_1}, f_{lsr_2}, \ldots, f_{lsr_{11}}]^T$.

Since we also want to integrate visual information while using the labels automatically generated from laser data, a correspondence between laser and vision data needs to be established. The visual features need to be correlated to the sensor designated as the base sensor, in our case the laser. This is done by projecting the laser returns onto the camera images, using the calibration procedure described by Zhang and Pless (2004). This is the first step in order to extract visual features from the images provided by a monocular color camera. In a second step, projected clusters are used to define regions of interest (ROI) in the camera images. Size of the ROIs is adjusted according to a simple heuristic that takes into account the mean range to the objects in the laser coordinate frame.

Our approach is based on the extraction of a large and generic set of features (**?**), in contrast to the other classic approach based on the extraction of a small class-specific set of features. In each of the ROI defined by the laser clusters, we extract a set of visual features f_{visual} of dimension 940. Each of them includes: texture information containing the steerable pyramid (Simoncelli and Freeman 1995), coefficients of the ROI, together with the minimum and maximum one, 3D histograms of the RGB and HSV channels of the ROI, Haar features (Viola and Jones 2004), Canny edges (Canny 1986), number of pixels detected as belonging to an edge, number of straight lines, SIFT features (Lowe 2004), pyramid histograms of oriented gradients (Bosch et al 2007), and maximally stable extremal regions (Matas et al 2002).

Table 2 Classification accuracy [mean/σ] (in %) for the supervised classifiers.

	Laser		Vision	
	UL	TL	UL	TL
Logitboost	93.37/1.37	97.37/0.69	92.95/0.83	95.18/1.06
SVM	94.51/0.86	96.02/0.66	91.47/1.94	92.95/0.47
LD	82.41/2.31	92.04/0.91	84.6/2	92.17/1.54
UL: training with unsupervised labels TL: training with true labels				
Clustering accuracy [mean/σ] = 96.22/0.83				

5.2 Supervised classifiers

This section provides a first empirical analysis of the proposed architecture which validates the use of unsupervised labels to train supervised classifiers. Based on the comparison of supervised classifiers presented by Caruana and Niculescu-Mizil (2006), we evaluate three classifiers with different characteristics: Logitboost, Support Vector Machines, and Linear Discriminant classifiers. These classifiers are introduced below.

Boosting algorithms incrementally combine "weak" base classifiers to produce an output that can be significantly better than the base ones. We use the Logitboost version of boosting (Friedman et al 2000), implemented with decision stumps (two-node decision trees) as these weak classifiers. One of the characteristics of Logitboost is that it is able to provide as the output the probabilities over the classes together with the estimated labels, that is $P(Q|f)$, where Q denote the classes, and f the features. This is done by considering a generalized additive model and logistic regression. Support Vector Machines (SVM) perform classification by finding the optimal margin with respect to the support vectors, defined as the data samples along the decision boundaries. In our experiments, we used the library provided by Chang and Lin (2008). Multi-class probability estimates can be obtained by using the SVM extension proposed by Wu et al (2003). Linear discriminants (LD) are very simple classifiers that use hyperplanes defined as decision boundaries in order to separate different regions of a high-dimensional input space (Duda et al 2001).

The first set of evaluations presented in this section aims at assessing which classifier (amongst SVMs, Logitboost and LD classifiers) performs the best, when the general architecture presented in Figure 1 is instantiated with a laser as Base Sensor and Sensor 1, and a monocular color camera as Sensor 2. During this initial testing phase, the various candidate classifiers are assigned to each sensor input in order to evaluate the classification performance. The performance of these classifiers is evaluated using unsupervised gener-

Table 3 Classification accuracy [mean/σ] (in %) for combined classifiers.

	Accuracy
SVM for laser (UL)	94.51/0.86
Logitboost for vision (UL)	92.95/0.83
Combined (UL)	96.49/0.5

Table 4 Computation times [mean/σ] (in secs). The times reported for the training stage correspond to total times for the entire dataset, whereas the times for the inference stage correspond to average times per data point.

	Training time	Inference time
Unsupervised labeling (EM)	25.18/9.87	
SVM for laser (UL)	0.44/0.03	0.02/0.01
Logitboost for vision (UL)	21.65/0.62	0.41/0.02

Table 5 Confusion matrix (in %) for the combined approach.

	Bike	Pedestrian	Car
Bike	88.72	3.8	7.49
Pedestrian	0.66	94.83	4.51
Car	1.83	1.09	97.08

ated labels (UL) for training. For comparison purposes, the evaluation is also performed training the classifiers with true labels (TL) obtained from manual annotation. In all these cases (and throughout this work), classification accuracies are always computed using the true labels (TL), regardless of whether the classifiers have been trained using true (TL) or unsupervised labels (UL). Test error was estimated using $n-$fold cross validation. Results are averaged over the n runs. In our initial validation we used a dataset containing 2286 laser and vision feature points, out of 171 tracks. Each of these points corresponds to a vector containing laser f_{lsr} and vision f_{vision} features, determined by laser clusters from tracks and their corresponding ROI in the images. In the remainder of this paper, we will refer to these laser and vision feature points as samples. The selection of these tracks is based on the procedure introduced in Section 4.1 such that the confidence that they correspond to actual dynamic objects is very high. Persistent tracks are identified first, and stamp features are extracted for unsupervised generation of labels. Using also these persistent tracks, samples for supervised classification are extracted. The classification involves the three classes bike, pedestrian and car. The datasets have been manually labeled for these classes for evaluation purposes.

Table 2 presents the classification results using five-fold cross validation, for Logitboost, SVM and LD classifiers, using laser and vision. Classification performance is shown using the mean and standard deviation (σ) of the accuracy. Training was computed using both track-based unsupervised generated labels (UL) and true labels (TL). For Logitboost, the learning process achieved reasonable convergence at 40 iterations. The best empirical results for SVM were obtained for a soft margin (C-SVC) con figuration (with $C = 1$) that allows some of the training points to be misclassified, and using an RBF exponential kernel of the form $e^{-\gamma|u-v|^2}$ with $\gamma = 1$. The track-based unsupervised generation of labels is computed within the five-fold cross validation procedure, reaching an average accuracy of 96.22% ($\sigma = 0.83$). Thanks to the very accurate track-based unsupervised labeling obtained using persistent stamps, accuracy of the supervised classifiers is very high.

5.3 Combined classification

Based on the previous analysis, appropriate classifiers for each modality are chosen. The classifiers with the best individual performance previously tested in Section 5.2 are now combined to obtain a final, unified output classification. SVM and Logitboost gave almost equal results, being SVM slightly better for laser and Logitboost for vision detection (Table 2 (grayed cells)). Therefore, since none of these two algorithms show a clear advantage over the other, we chose to combine the SVM result obtained with laser, and the Logitboost result obtained with vision.

The combination of classifiers is achieved using Bayesian averaging (Hoeting et al 1999; Bishop 2006). Bayesian averaging uses the posterior probabilities given by different classifiers. In our case, these are directly provided by Logitboost, and can be also obtained in SVM (see Section 5.2). Bayesian averaging combines K different models as:

$$P(Q|f) = \sum_{k=1}^{K} P(Q|M_k, f)P(M_k|f), \tag{10}$$

where $P(Q|M_k, f)$ is the posterior for the models, and $P(M_k|f)$ the posterior model probability, for each model M_k. In our implementation, $P(Q|M_k, f)$ is the output from both classifiers, SVM for laser (M_1), and Logitboost for vision (M_2). The model probability $P(M_k|f)$ is assumed constant for both models, and is learned a priori by counting the distribution over the classes from the small annotated dataset used in Section 4.4. As before, the performance is evaluated using 2286 samples from 171 persistent tracks, and five-fold cross validation.

Table 3 shows the classification result obtained in this experiment, together with the accuracies previously obtained for the individual classifiers. The combined approach improves the performance of the individual classifiers, reaching an accuracy of 96.49%, reducing the standard deviation

Table 6 Precision and recall [mean/σ] (in %) for the combined approach.

	Bike	Pedestrian	Car
Precision	82.15/18.63	94.85/1.1	97.08/0.21
Recall	96.86/1.44	92.04/13.69	89.5/8.11

to almost half of the values for the separate classifiers. The computation times involved in the stages of the combined classification are shown in Table 4. The times reported for the training stage correspond to total times for the entire training sets, whereas the times for the inference stage correspond to average times per data point. A confusion matrix is also used here (and throughout this work) to show the performance in the classification. In a confusion matrix rows denote real (ground truth) classes and columns refer to the estimated classes. The corresponding confusion matrix indicating the accuracy (in %) for the classes bike, pedestrian and car for the combined approach is shown in Table 5. This confusion matrix shows a strong diagonal, with highly accurate results, particularly for the classes pedestrian and car. The lower accuracy in the class bike (although still accurate) compared to pedestrian and car is due to the unbalanced representation of the classes in the dataset. In fact, the percentage of samples belonging to this class is quite small (only 6%). This leads to a smaller number of training samples of this class being selected (with respect to the other two classes) in the cross validation procedure, which then induces a slightly inferior performance. Another representation of this confusion matrix is shown through precision and recall values presented for the combined approach in Table 6.

6 Experimental evaluation

This section presents experiments showing the performance of the proposed architecture. There are two main goals involved in the experiments. Firstly, the *generalization* of the system is evaluated. Initial experiments in this section use the information of all the incoming tracks, that is all the "unpersistent" tracks received, without limiting classification to persistent tracks and their samples. Secondly, the *adaptive* capabilities are explored. This is done by performing on-line retraining through the self-supervised scheme.

The datasets were obtained in an outdoor, urban environment at the University of Sydney campus, using the system provided by ACFR et al (2006). They consist of a sequence of images temporally correlated with laser scans. The images were provided by an IDS camera of resolution 1280×1024 at 5Hz, whereas the laser scans were obtained by a Sick scanner LMS291 in high speed mode (500kBps). The speed of the vehicle varied between 0-40km/h during 60 minutes of logged data. As mentioned earlier, the algorithmic procedure presented by Zhang and Pless (2004) was used to compute the camera-laser calibration. Datasets consist of 364 tracks, generating 2822 samples. The number of classes for classification is again three, for the classes bike, pedestrian and car. Datasets have been annotated accordingly for evaluation purposes. We use in these experiments Logitboost for classification of vision features (40 iterations), and SVM for laser features (RBF kernel, with $\gamma = 1$ and $C = 1$). Combined classification integrates these two separate classifiers using Bayesian averaging. Five-fold cross validation procedure is used in all the experiments.

Table 7 Confusion matrix (in %) for combined approach using general samples. Classification accuracy of 95.78% ($\sigma = 0.09$).

	Bike	Pedestrian	Car
Bike	77.14	5.71	17.14
Pedestrian	0.55	94.13	5.32
Car	2.04	1.43	96.53

Table 8 Precision and recall [mean/σ] (in %) for combined approach using general samples.

		Bike	Pedestrian	Car
Prec.	All features	71.11/10.08	94.12/1.24	96.53/0.67
	No speed feats.	26.33/14	92.29/1.85	95.31/0.79
Rec.	All features	96.64/0.7	91.14/10.44	82.32/8.42
	No speed feats.	75.67/37.84	59.83/4.79	83.01/9.79

6.1 Classification of general samples

The experiments shown in previous sections for validating the self-supervised architecture have been properly evaluated in the sense that the training data subsets are completely disjoint from the testing subsets. However, the evaluation has been constrained to persistent tracks, and to their corresponding persistent samples. In the following experiment, rather than processing only the tracks identified as persistent, we extract and process all samples.

Effectively, we perform classification of general samples derived from all tracks. Persistency of the tracks is still evaluated to allow an accurate unsupervised generation of labels as described in Section 5. Stamp-based unsupervised labels are used to train SVM and Logitboost classifiers. However, inference is performed using general samples, that is all the samples corresponding not only to persistent tracks but also to unpersistent tracks (tracks that might not correspond to actual moving obstacles). Using the full set of laser features (that is, all features in f_{lsr} of size 11) the combined classification reaches an accuracy of 95.78% ($\sigma = 0.09$), for the confusion matrix shown in Table 7. In fact, this performance

almost equals the high accuracy for combined classification using persistent samples only, as presented in Table 3 (grayed cell).

Precision and recall values are shown for this case in Table 8, for the first and third ("all features") rows. When removing speed features from the laser feature vector $f_{lsr} = [f_{lsr_1}, f_{lsr_2}, \ldots, f_{lsr_9}]^T$, the combined classification accuracy reaches 94.46% ($\sigma = 0.84$), achieving also in this case an excellent "track independent", on-the-spot classification. Second and fourth rows rows in Table 8 present the precision and recall values for this track independent case. As can be seen, the classification performance remains very high when using general samples. The use of instantaneous information only without considering track speed related features does not greatly affect the accuracy of the self-supervised classification. However, speed features help to differentiate some of the classes. The removal of these features is detrimental for the classes bike and pedestrian, as illustrated in the precision and recall results in Table 8.

Inference times in this experiment are similar to the ones shown in Table 4. As presented, the total inference time is below 0.5 secs using non-optimized Matlab routines, suggesting that the system is able to perform real-time classification of general samples in an equivalent optimized architecture.

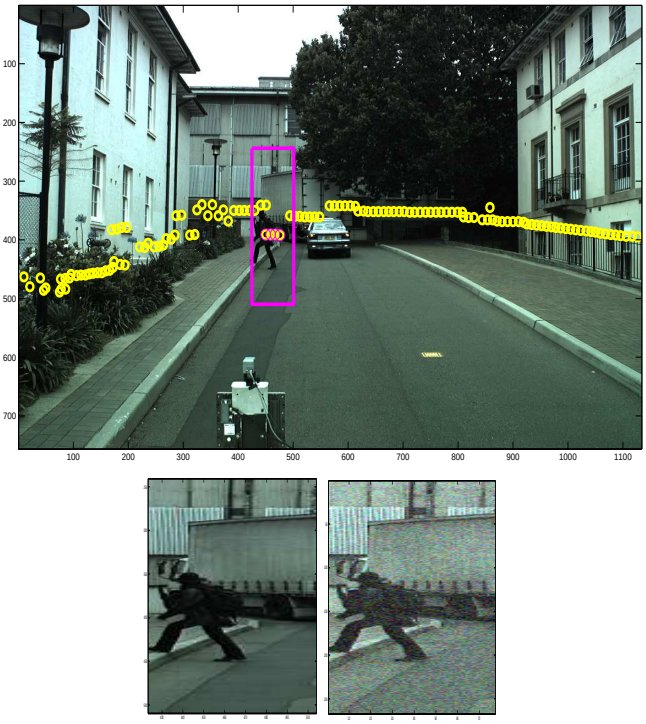


Fig. 7 Perturbation of the visual sensor. Additive random noise is applied to each of the RGB channels in the ROI images, as shown on the bottom images for the ROI corresponding to the detected pedestrian.

6.2 Classification of corrupted data

This section evaluates the advantages of having a self-supervised scheme with an unsupervised module in the architecture, such that adaptive tuning of the classifiers can be done on-the-fly. One way to assess this is by analyzing if the retraining capability, for instance, is able to cope with malfunctioning behavior in one of the sensors. This is done, in our case, by applying a perturbation to one of the sensors (camera) and evaluating its performance. We specifically apply additive i.i.d. random noise with unit variance to each of the RGB channels in the ROI images, prior to obtaining the visual features. This is illustrated in the example shown in Figure 7 for the ROI that corresponds to a pedestrian.

The evaluation is undertaken adding different proportions of corrupted images (between 0% and 100%) to the datasets, for the following two scenarios:
(i) Standard supervised approach. This setup replicates the case where no unsupervised module is present in the system, and the working conditions suddenly change. We train the vision supervised classifier with a training set that is not corrupted, and the true labels. Inference is then performed using corrupted sets. Note that since this approach relies on provided labels and can only be trained once, it is not able to re-train on-the-fly.
(ii) Self-supervised scheme. In this case the self-supervised architecture proposed in this paper is used. Training is performed using sets of corrupted data, and track-based unsupervised generated labels. Inference is then performed using corrupted sets. Training the system with corrupted data simulates on-line retraining of the classifier, which is possible by the self-supervision capabilities of the system.

Figure 8 shows the results obtained for these two different setups. The plots indicate the accuracy for each method for different levels of corrupted data. As can be seen, the supervised scenario is initially more accurate than the self-supervised because of the use of true labels (TL) for training. However, as the amount of corrupted data increases (level of noise), the accuracy of the supervised approach is notably affected by this demanding scenario. The self-supervised approach, on the other hand, is able to adapt on-line thanks to the unsupervised module performing label generation. The self-supervision capabilities allow the system to maintain a high classification accuracy above 90%.

Computation times in Table 4 show that the total training time is below 50 seconds. This indicates the feasibility of the architecture for on-line retraining, which can be regularly performed in an efficient manner due to the reduced training times.

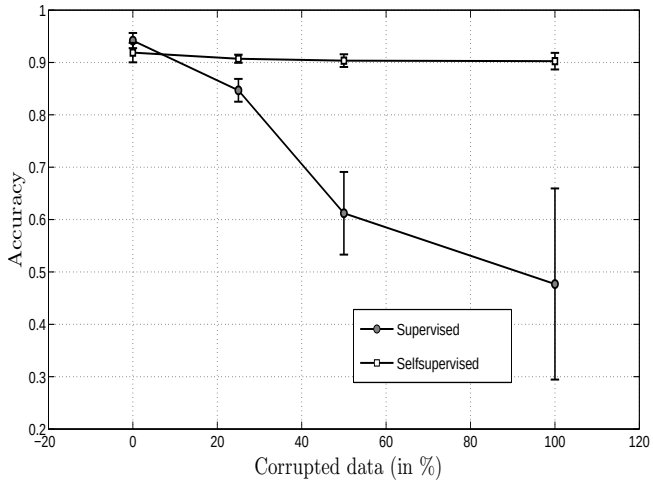


Fig. 8 Vision classification accuracy (in %) for two setups using general samples: Supervised (i) and Self-supervised scheme (ii). The accuracy (with error bars indicating standard deviation) is shown for both approaches using different levels of corrupted data.

7 Conclusions and future work

This paper introduced a self-supervised multi-sensor architecture to perform automatic moving obstacle classification. The approach relies on obtaining persistent and discriminative features from a sensor. Unsupervised clustering is used to achieve robust automatic labeling, and this information is then used to train a set of supervised classifiers with the aim of achieving instantaneous classification.

The introduction of laser stamp allows us to derive a highly accurate procedure for unsupervised generation of labels. The integration of this labeling methodology within a self-supervised learning scheme provides our architecture with additional capabilities such as generalization and adaptability. Experiments show that the system performs robustly, obtaining accurate individual and combined classification rates for the three classes bike, pedestrian and car. The system is able to generalize well, achieving high classification rates for general samples, based on the training provided by persistent information. The self-supervised scheme presents advantages compared to supervised approaches, in terms of adaptability obtained with on-line retuning capabilities.

The architecture evaluated in this work assumed that the number of classes was known for the automatic generation of labels based on clustering. This seems to be a valid assumption considering that a priori knowledge can include, for instance, most of the possible classes for moving objects present in urban environments. Available methods, however, can perform clustering using variable or no predefined number of classes (e.g., VBEM Attias (1999)). We therefore intend to evaluate these approaches to further assess if they could provide more adaptability and accuracy to the self-supervised system.

8 Acknowledgments

The authors would like to thank Oliver Frank for providing the multiple target tracking code. This work is supported by the Australian Research Council (ARC) Centre of Excellence program and the New South Wales Government.

References

ACFR, of Sydney TU, LCR, del Sur UN (2006) PAATV/UTE Projects. Technical Report ACFR

Attias H (1999) Inferring Parameters and Structure of Latent Variable Models by Variational Bayes. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco CA

Besl P, McKay N (1992) A Method for Registration of 3-D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2):239–256

Bishop CM (2006) Pattern Recognition and Machine Learning. Springer

Bosch A, Zisserman A, Munoz X (2007) Representing Shape with a Spatial Pyramid Kernel. In: CIVR '07: Proceedings of the 6^{th} ACM International Conference on Image and Video Retrieval, ACM, Amsterdam, The Netherlands, pp 401–408

Brooks CA, Iagnemma KD (2007) Self-Supervised Classification for Planetary Rover Terrain Sensing. In: 2007 IEEE Aerospace Conference, IEEE, Big Sky, Montana

Canny JF (1986) A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6):679–698

Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In: ICML '06: Proceedings of the 23^{rd} International Conference on Machine Learning, ACM, New York

Chang C, Lin C (2008) A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Cohen I, Cozman FG, Bronstein A (2002) The Effect of Unlabeled Data on Generative Classiffiers, with Application to Model Selection. HP Laboratories Palo Alto

Dahlkamp H, Kaehler A, Stavens D, Thrun S, Bradski G (2006) Self-supervised Monocular Road Detection in Desert Terrain. In: Robotics: Science and Systems, MIT Press

DARPA DARPA (2007) DARPA Urban Challenge. http://www.darpa.mil/grandchallenge/

Dempster A, Laird NM, Rubin DB (1977) Maximum Likelihood from Incomplete Data Via the EM Algorithm. Journal of the Royal Statistical Society 39(1):1–38

Duda R, Hart P, Stork D (2001) Pattern Classification. John-Wiley, New York

Elfes A (1989) Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University

Frank O, Nieto J, Guivant J, Scheding S (2003) Multiple Target Tracking using Sequential Monte Carlo Methods and Statistical Data Association. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Las Vegas, USA

Friedman JH, Hastie T, Tibshirani R (2000) Additive Logistic Regression: a Statistical View of Boosting. Annals of Statistics 28(2):337–374

Hoeting J, Madigan D, Raftery AE, Volinsky CT (1999) Bayesian Model Averaging: A Tutorial. Statistical Science 14(4):382–401

Katz R, Nieto J, Nebot E (2008) Probabilistic Scheme for Laser Based Motion Detection. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Nice, France, pp 161–166

Lowe D (2004) Discriminative Image Features from Scale-invariant Keypoints. International Journal of Computer Vision 60(2):91–110

Luber M, Arras KO, Plagemann C, Burgard W (2008) Classifying Dynamic Objects: An Unsupervised Learning Approach. In: Robotics: Science and Systems, MIT Press

Luo J, Savakis AE (2001) Self-supervised Texture Segmentation using Complementary Types of Features. Pattern Recognition 34(11):2071–2082

Matas J, Chum O, Urban M, Pajdla T (2002) Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In: British Machine Vision Conference 2002, British Machine Vision Association

McLachlan G, Krishnan T (1997) The EM Algorithm and Extensions. Wiley Series in Probability and Statistics

Moravec H, Elfes A (1985) High Resolution Maps from Wide Angle Sonar. In: International Conference on Robotics and Automation, IEEE

Ng A, Jordan M, Weiss Y (2001) On Spectral Clustering: Analysis and an Algorithm. In: NIPS: Advances in Neural Information Processing Systems 14

Rabiner LR (1990) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: Waibel A, Lee K (eds) Readings in Speech Recognition, Morgan Kaufmann Publishers Inc., pp 267–296

Schultz D (2006) A Probabilistic Exemplar Approach to Combine Laser and Vision for Person Tracking. In: Robotics: Science and Systems, MIT Press

Simoncelli E, Freeman W (1995) The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation. International Conference on Image Processing 3:444–447

Stavens D, Thrun S (2006) A Self-supervised Terrain Roughness Estimator for Off-road Autonomous Driving. In: Conference on Uncertainty in AI (UAI), Cambridge, MA

Sun Z, Bebis G, Miller R (2006) On-Road Vehicle Detection: A Review. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(5):694–711

Torralba A, Murphy K, Freeman WT, Rubin MA (2003) Context-based Vision System for Place and Object Recognition. In: ICCV '03: Proceedings of the 2003 ICCV International Conference on Computer Vision, IEEE, Nice, France

Tsuchiya M, Fujiyoshi H (2006) Evaluating Feature Importance for Object Classification in Visual Surveillance. In: ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition, IEEE Computer Society, Washington, DC, pp 978–981

Viola P, Jones M (2004) Robust Real-time Object Detection. International Journal of Computer Vision 57:2

Weber M, Welling M, Perona P (2000) Towards Automatic Discovery of Categories. In: CVPR '00: Proceedings of the 2000 Conference on Computer Vision and Pattern Recognition (CVPR '00), IEEE Computer Society, Hilton Head Island, South Carolina

Wu T, Lin C, Weng RC (2003) Probability Estimates for Multi-class Classification by Pairwise Coupling. Journal of Machine Learning Research 5:975–1005

Zelnik-Manor L, Perona P (2004) Self Tuning Spectral Clustering. In: NIPS: Advances in Neural Information Processing Systems 17

Zhang Q, Pless R (2004) Extrinsic Calibration for a Camera and Laser Ranger Finder (Improves Camera Intrinsic Calibration). In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Japan