Article

The International Journal of Robotics Research
2015, Vol. 34(4-5) 559–581

DOI: 10.1177/0278364914566514
ijr.sagepub.com

Online self-supervised learning for dynamic object segmentation

Vitor Guizilini and Fabio Ramos

Abstract

This paper proposes a novel technique for the automatic segmentation of dynamic objects, solely using information from a single uncalibrated moving camera and without the need for manual labeling (or any human intervention, for that matter). Matching pairs of sparse features are extracted from subsequent frames, and the resulting optical flow information is divided into two classes (static or dynamic) using the RANSAC algorithm. This initial classification is then used to incrementally train a Gaussian process (GP) classifier that is then able to segment dynamic objects in new images. The GP hyperparameters are optimized online during navigation, with new data being gradually incorporated into the non-parametric model as it becomes available while redundant data is discarded, to maintain a near-constant computational cost. The result is a vector containing the probability that each pixel in the image belongs to a dynamic object, along with the corresponding uncertainty estimate of this classification. Experiments conducted using different robotic platforms, ranging from modified cars (driving at speeds of up to 50 km/h) to portable cameras (with a full six-degree-of-freedom range of motion), show promising results even in highly unstructured environments with cars, buses and pedestrians as dynamic objects. We also show how it is possible to cluster individual dynamic pixels into different object instances, and then further cluster those into semantically meaningful categories without any prior knowledge of the environment. Finally, we provide visual odometry results that testify to the proposed algorithm's ability to correctly segment (and then remove) dynamic objects from a scene, and how this translates into a more accurate motion estimate between frames.

Keywords

Computer vision, dynamic objects, object segmentation, machine learning, Gaussian processes, online learning

1. Introduction

A truly autonomous robot requires precise knowledge of the environment around it in order to perform high-level tasks such as path planning (Szoke et al., 2009), obstacle avoidance (Nedevschi et al., 2004) and goal-oriented navigation (Cheng and Zelinsky, 1998). The most common way to address the problem of building a representation of the environment around a robot is by generating a map which contains the structures the robot will interact with during navigation. However, the iterative nature of building a map usually constitutes a challenge when dealing with dynamic objects, since their position varies with time and therefore cannot be estimated simply by continuous observation. Because of their unpredictability and overall importance in a scene, segmenting dynamic objects from a static background is an important step in applications such as collision warning and avoidance, surveillance, video mining, driver assistant systems and tracking. Of all sensors, cameras are becoming increasingly popular because they are relatively inexpensive, small, information-rich, of easy installation and have a wide field of view both horizontally and vertically. The texture and color information provided by visual sensors is also invaluable in situations where a better characterization of objects is necessary, such as road segmentation, lane detection and multi-tracking.

Machine learning techniques have been widely utilized in computer vision applications (Sheikh and Shah, 2005; Wang et al., 2011), and are known to provide robust models based on scarce and noisy datasets. This characteristic makes them especially attractive in real-world scenarios, where the same object may present itself in a wide variety of ways due to changes in scale, luminosity, viewpoint and even shape. By optimizing a non-parametric model over a large subset of examples (which could be obtained as a

Australian Centre for Field Robotics, School of Information Technologies, University of Sydney, Australia

Corresponding author:
Vitor Guizilini, Australian Centre for Field Robotics, School of Information Technologies, J12, University of Sydney, 1 Cleveland Street, Darlington NSW 2008, Australia.
Email: v.guizilini@acfr.usyd.edu.au

batch prior to the beginning of the navigation or during it, in an online fashion), these techniques are capable of generalizing over small variations in the object's appearance and extracting the properties that truly represent it, in a way that classical parametric models greatly struggle with.

The main goal of this paper is to provide a framework capable of segmenting dynamic objects from a static background using only a single uncalibrated camera placed on top of a mobile platform, without the need for extra sensors or any prior knowledge of the environment. We introduce a novel self-supervised algorithm based on Gaussian processes (GPs) (Rasmussen and Williams, 2006), a non-parametric Bayesian technique used here to estimate the probability that each pixel in an image belongs to a dynamic object, based on optical flow information obtained from sparse features matched between frames. The extraction of dynamic objects based on the consistency of the motion field has already been explored in the literature (Ibrahim et al., 2010; Han and DeSouza, 2011; Teutsch and Kruger, 2012), and here we expand on this notion from an unsupervised machine learning perspective. An initial classification is performed using the 7-point RANSAC algorithm (Fischler and Bolles, 1981), and the results are used as input for the GP to perform a second classification on the entire image, along with the corresponding uncertainty estimates. We show that this second layer of classification is capable of greatly improving segmentation results, by incorporating information from previous frames and thus allowing the system to gradually learn the behavior of structures around the vehicle. The GP hyperparameters are optimized online during navigation as new data becomes available, and the current non-parametric model is constantly updated by adding relevant information and removing redundant information, to keep the computational cost roughly constant.

Once this segmentation is complete, the results can be readily used in a wide range of applications. In particular, we introduce here a novel technique that is capable of clustering individual dynamic pixels into different object instances, based on a combination of optical flow, spatial coordinates and color/texture information. These various object instances, in turn, can also be clustered themselves, creating different object categories, and we show here that these categories contain semantically meaningful information even when no prior knowledge of the environment is available. In addition, we provide visual odometry results that testify to the proposed algorithm's ability to correctly segment (and then remove) dynamic objects from a scene, and how this translates into a more accurate motion estimate between frames.

The rest of this paper is divided as follows: Section 2 provides a brief overview of different methods and techniques for the segmentation of dynamic objects already available in the literature. We then move on to Section 3, where the proposed algorithm is introduced and its various stages are discussed, along with how they connect to each other from input (two subsequent frames) to output (dense pixel classification between static/dynamic classes). The next sections, 4 and 5, are devoted to explaining the theory behind each of these steps, the former focusing on the extraction of image information and the latter on GP classification. The various experiments conducted to test the proposed algorithm are presented and discussed in Section 6, and Section 7 concludes and presents future research directions.

2. Related work

Several applications of dynamic object detection assume a static camera, which implies that any non-dynamic object will maintain its position over time. In this scenario it is possible to statistically model the background, essentially 'filtering it out', and treat any change in pixel intensity as a potential dynamic object. In Wren et al. (1997) each pixel of the image is modeled as a Gaussian distribution, whose parameters are learned from observations in consecutive frames, and in Koller et al. (1994) a Kalman filter is used in a similar fashion. When a uni-modal solution is ill-suited (e.g. when the background changes in a predictable manner, such as trees swaying, fans rotating or water flowing), a mixture of Gaussian models has been applied with satisfactory results (Friedman and Russell, 1997; Stauffer and Grimson, 2000a; Ellis and Xu, 2001), and in Stauffer and Grimson (2000b) and Stenger et al. (2000) a hidden Markov model (HMM) is used to model the background while exploiting spacial dependencies between pixels. Other approaches forego pixel-wise locality in favor of regional models of intensity, such as eigenvalue decomposition (Oliver et al., 2000) and autoregressive moving averages (Monnet et al., 2003; Zhong and Sclaroff, 2003). A mixture of local and regional models is employed in Toyama et al. (1999), and in Sheikh and Shah (2005) a foreground model is explicitly maintained in order to improve the detection of dynamic objects without using tracking information.

In other applications, however, the visual sensor is mobile, usually mounted on top of a robotic platform. In this scenario it is impossible to separate background and foreground solely by tracking pixel intensity changes, as static objects will also experience relative motion due to camera rotation and translation between frames. A straightforward way of segmenting this sort of image is to model the ground plane and treat everything else as an object (Zhao and Thorpe, 2000; Nedevschi et al., 2004; Soga et al., 2005), however, this approach tends to fail in crowded environments where the ground plane is not readily visible. A weaker ground plane constraint is presented in Ess et al. (2009), where a coupling between object detection and scene geometry is maintained using a Bayesian network.

If a significant portion of the environment is assumed to be static, the relative motion of static objects can be filtered out by calculating the optical flow (Horn and Schunck, 1980; Namdev et al., 2012) of the image and using a voting method, such as RANSAC (Fischler and Bolles, 1981),

to elect the most probable motion hypothesis. Any region that does not comply with this constraint is assumed to be dynamic, and can be tracked using classical approaches such as extended Kalman filters (Welch and Bishop, 1995) or particle filters (Van der Merwe et al., 2001) (robust data association algorithms (Cox, 1993) and occlusion-handling techniques (Ellis and Xu, 2001; Ess et al., 2009) are necessary for dealing with very cluttered environments). If more than one camera is available, a stereo triangulation can provide a 3D position estimate for matched features (Hartley and Zisserman, 2004), incorporating extra information that could be used to facilitate and improve object clustering and tracking (Taluker and Matthies, 2004; Almanza-Ojeda and Ibarra-Manzano, 2011). More accurate detection can also be achieved by applying category-specific models to separate the static background from already established dynamic objects, either on a 3D point-cloud (Keck Jr et al., 2006; Arras et al., 2007), directly on the camera images (Dalal and Triggs, 2005; Leibe et al., 2008) or in a combination of both (Ess et al., 2007; Spinello et al., 2008). The static background information can also readily be used to improve visual odometry applications (Nister et al., 2006; Guizilini and Ramos, 2012, Bak et al., 2014) since its optical flow values now solely reflect the camera's own rotation and translation.

Due to a steady increase in computational power and storage capacity, many researchers are now moving away from increasingly complex parametric models, designed for a specific task based on a deep understanding of the process at hand, and focusing instead on machine learning techniques (Bishop, 2006; Rasmussen and Williams, 2006). These techniques eliminate the need for an explicit model of the underlying phenomenon by using training data, containing examples of the problem it is trying to solve, to learn the direct transformation between inputs and outputs. In computer vision specifically, machine learning techniques are beneficial due to the high-level dense nature of image information, which poses a challenge to traditional parametric approaches. By allowing the system to learn the underlying mapping between inputs and outputs, instead of explicitly defining it, we may discover patterns and correlations that are not readily apparent and can provide further insight into the nature of the task. Examples of computer vision applications that have been addressed using machine learning techniques include face recognition (Guo et al., 2000), people detection and tracking (Spinello et al., 2008), motion analysis (Wang et al., 2011), visual odometry (Guizilini and Ramos, 2012), scene classification (Maron and Ratan, 1998), 3D reconstruction (Saxena et al., 2009) and denoising (Hammond and Simoncelli, 2007).

Learning strategies can be broadly divided into two categories, according to how available information is incorporated into the non-parametric model. Batch algorithms have access to all training data before any inference is done, and so all available information can be incorporated simultaneously. Online algorithms receive new data during the inference process, and so new information has to be gradually incorporated into the non-parametric model as it becomes available. While batch algorithms are usually more robust, since the simultaneous processing of all information allows for a better understanding of the underlying phenomena, the resulting model is incapable of further learning new patterns and behaviors. Online algorithms, on the other hand, provide an iterative learning framework that never ceases to refine the current model, adding relevant information while removing redundant information. For this reason, most current approaches rely on a combination of both, with an initial model being generated using batch training data and then further refined in an online fashion using new data collected during inference. In Dekel (2008) a technique for online-to-batch conversion is presented which maintains online efficiency in a batch environment. Kivinen et al. (2003) explores an online extension to support vector machines (SVMs) (Cristianini and Shawe-Taylor, 2000), discussing the kernel trick and the value of large margins in classification. The incorporation of sparseness into an online algorithm is presented in Csat and Opper (2002) and Ranganathan and Yang (2008) as a way to address large datasets in a timely manner. A technique for online boosting is described in Yang and Belongie (2009), for both regression and classification scenarios.

Another way of dividing different learning strategies is related to the nature of information available to be incorporated into the non-parametric model. Supervised approaches (Vapnik, 1995; Mohri et al., 2012) require ground-truth values for all observed examples of the underlying phenomenon, be it from another independent sensor or provided manually, by hand-labeling. Semi-supervised approaches (Chapelle et al., 2006; Zhu, 2006) are initialized from a supervised standpoint, with a finite set of observations paired with their respective ground-truth, and this initial model is then incrementally refined by incorporating new unlabeled observations. Finally, unsupervised approaches (Hinton and Sejnowski, 1999; Ghahramani, 2004) completely eliminate the need for ground-truth, relying instead solely on unlabeled data to generate a model of the underlying phenomenon. Because there is no ground-truth, it is impossible to generate a penalty/reward signal to evaluate a potential solution, which arguably makes this branch of online learning strategies the hardest one to solve. A possible variation of unsupervised learning is called self-supervised (Sofman et al., 2006), because it is capable of self-generating ground-truth estimates during the training process, thus enabling the use of supervised techniques in an unsupervised manner.

We propose here a novel algorithm for the automatic segmentation of dynamic objects that is self-supervised, which means that it does not require any prior knowledge of the environment surrounding the camera. There is no human intervention at any stage of the proposed algorithm, and no other sensor is required to provide ground-truth for the segmentation process. This approach is particularly useful because it can be readily applied to any sort of environment, by allowing the system to gradually learn different

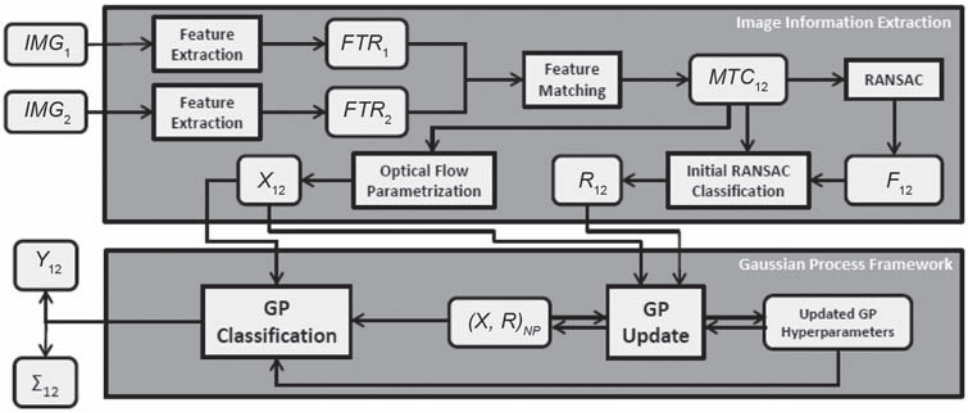

Fig. 1. Diagram of the proposed algorithm for the segmentation of dynamic objects.

patterns and react to gradual changes in structure behavior. Also, the proposed algorithm is robust to camera motion, being able to filter out ego-motion and retain only optical flow information that is created by external movement (e.g. dynamic objects). This is still an open problem in computer vision, and allows the use of the proposed algorithm in a much wider range of applications.

3. Algorithm overview

A diagram of the proposed algorithm for the self-supervised segmentation of dynamic objects is depicted in Figure 1. First of all, we can see that it receives as input a pair of frames, IMG_1 and IMG_2, and returns as output a vector Y_{12} containing the dense pixel classification between the static/dynamic classes, along with the corresponding uncertainty estimates Σ_{12}. The algorithm can roughly be divided into two stages: *image information extraction* and *Gaussian process framework*, the former dealing with the extraction of information from the input images and the latter dealing with the GP classification of such information.

The first stage, image information extraction, starts by generating the feature sets FTR_1 and FTR_2 from the input images. These feature sets are then matched to generate the matching set MTC_{12}, which constitutes the sparse optical flow used to determine motion in different portions of the environment around the vehicle. The RANSAC algorithm (Fischler and Bolles, 1981) is then used, in conjunction with this matching set, to calculate the fundamental matrix F_{12} between frames, which contains the geometric constraints that govern camera motion. Assuming that most of the environment is static (an assumption that will be further discussed later on), it is possible to calculate the most probable motion hypothesis for the camera, that solely reflects its translation and rotation between frames. Any matching pair that does not comply with these constraints will, therefore, belong to a dynamic object, and this information provides the initial classification R_{12} between static/dynamic features. Additionally, the matching set also undergoes a parametrization process, which generates the input vector X_{12} used by the GP framework.

The resulting pair $(X, R)_{12}$ moves on to the second stage, Gaussian process framework, where it is decided whether or not this information is relevant (and should therefore be incorporated into the non-parametric model $(X, R)_{NP}$), or redundant (and should therefore be discarded). This non-parametric model is empty at the beginning of navigation and the hyperparameters are selected randomly, indicating that no prior knowledge of the environment is necessary. If these hyperparameters are available (e.g. from a previous run) they can be incorporated seamlessly. The GP hyperparameters are then updated according to this new information, and the resulting optimized hyperparameters are used in conjunction with the non-parametric model to generate the final classification Y_{12} for each matching pair, along with the corresponding uncertainty estimate Σ_{12}. Since this final classification does not require any optical flow information, only image coordinates and color intensities, it can be performed equally in any portion of the image, thus allowing a dense classification that covers even featureless portions of the scene. The result is a $2 \times h \times w$ vector (where h and w are, respectively, the image's height and width resolution) containing values in the range $[0, 1]$ that indicate the probability that each pixel in the image belongs to a dynamic object, along with the corresponding uncertainty with regard to that estimate.

4. Image information extraction

The approach proposed here uses sparse optical flow information from a pair of consecutive frames obtained using a single uncalibrated camera configuration. A histogram filter was applied to each frame to account for global luminosity changes and allow for a proper representation of dark areas. The initial feature extraction is performed using a combination of both the scale-invariant feature transformation (SIFT) (Lowe, 2004) and the Shi–Tomasi corner detection (Tomasi and Tomasi, 1994) algorithms, with sub-pixel

accuracy and frame-to-frame tracking, to ensure a high density of features throughout the entire image (Figure 2(a)). Empirical tests show that the invariance properties of SIFT features ensure robustness during the matching process, whereas the Shi–Tomasi corner detector is particularly suitable for tracking over a series of frames. Any other similar method could be readily applied, both for speed purposes (Bay et al., 2006) or as a way to model different environment properties. The feature set from each frame is then matched with the feature set from the subsequent frame using 128-element SIFT descriptors, to generate the optical flow information that determines motion in different portions of the environment around the vehicle. The resulting set is shown in Figure 2(b), where each match is represented as a red line segment connecting the corresponding features from each frame.

This optical flow information is then used in two different ways: the *initial RANSAC classification* and the *optical flow parametrization*. The former produces the initial classification between the static and dynamic classes that serves as ground-truth for the GP framework, while the latter produces the vectors that also serve as input for the GP framework. The next two sections describe these two stages in further detail, introducing the techniques utilized and discussing these intermediate results.

4.1. Initial RANSAC classification

The initial RANSAC classification stage provides the ground-truth used by the GP framework, thus eliminating the need for manual labeling or extra sensors that are capable of providing such information directly. The RANdom SAmple Consensus (Fischler and Bolles, 1981) is an iterative algorithm used to estimate the parameters of a mathematical model from a set of observed data which contains outliers. For the application at hand, the mathematical model is the fundamental matrix F that encodes the geometric constraints correlating the visual system between frames (Hartley and Zisserman, 2004), and the outliers are the dynamic objects in the environment, which generate optical flow that cannot be explained away by the camera's own translation and rotation. The RANSAC algorithm basically elects a random sample from the available data, builds a model and then tests all remaining data points against this model, and the model with the highest number of inliers is determined to be correct. If most of the environment is considered static, it is natural to expect that RANSAC should elect the model that best represents a static environment, and therefore any matching pair that does not comply with these constraints should belong to a dynamic object. If most of the environment is indeed not static, this hypothesis does not hold and the algorithm will start to converge to a different motion model. We assume here that any violations of this hypothesis will be temporary, and the learning aspect of the proposed algorithm should allow it to recover from local failures.

We can see in Figure 2(b) that there is a substantial number of false matches, mostly due to similarities in different portions of the image, noise or occlusion. It is therefore necessary to make a distinction between these false matches (the real outliers) and dynamic features, that were correctly matched but have an optical flow different than that generated by static features. We can also see in Figure 2(b) that, even though a large portion of the environment is dynamic (especially in the right column), there are still enough static features to ensure that RANSAC will converge to a model that represents a static environment, mostly in the street and in the upper portions of the image. Once this model is obtained, each matching pair is tested against it and a measurement of error y_n is calculated, based on the Euclidean distance between the match and its corresponding epipolar line (Figure 3). If the match falls precisely on the epipolar line, the projection error is zero and its optical flow is consistent with that of a static object, and the higher the error is the further away the match is from the epipolar line. This error estimate provides a metric for the determination of which features are dynamic and which are static, and matching pairs with a projection error above a certain threshold are discarded as outliers. This threshold is also used to normalize the remaining projection errors to values ranging in $[0, 1]$, which indicate the probability that each matching pair belongs to a dynamic object.

Examples of this initial RANSAC classification are shown in Figure 4, where the projection error values were converted to colors ranging from red (static objects) to green (dynamic objects). In Figure 4(a) the vehicle is moving forward, generating a relative optical flow component that also influences static features. Because of that, even though most dynamic objects were correctly classified, there are several portions of the image that are wrongly classified as dynamic, especially towards the border where the relative optical flow component of camera motion is stronger. The street also contains several features that were wrongly classified, mostly due to the lack of texture that increases ambiguity during the matching process and with it the chances of generating outliers. In Figure 4(b) the vehicle is not moving, and so any optical flow detected is solely due to the presence of dynamic objects. It is clear that this scenario greatly facilitates RANSAC classification, as now any feature that experiences motion between frames can be safely considered dynamic. However, we can also see several dynamic objects that are not represented by any features, and thus are not detected by the algorithm. We attribute this lack of representation to occlusion, the presence of deformable objects and local luminosity changes.

4.2. Optical flow parametrization

Even though the SIFT descriptor is highly effective in feature matching between frames, possessing several invariance properties that promote a unique correspondence

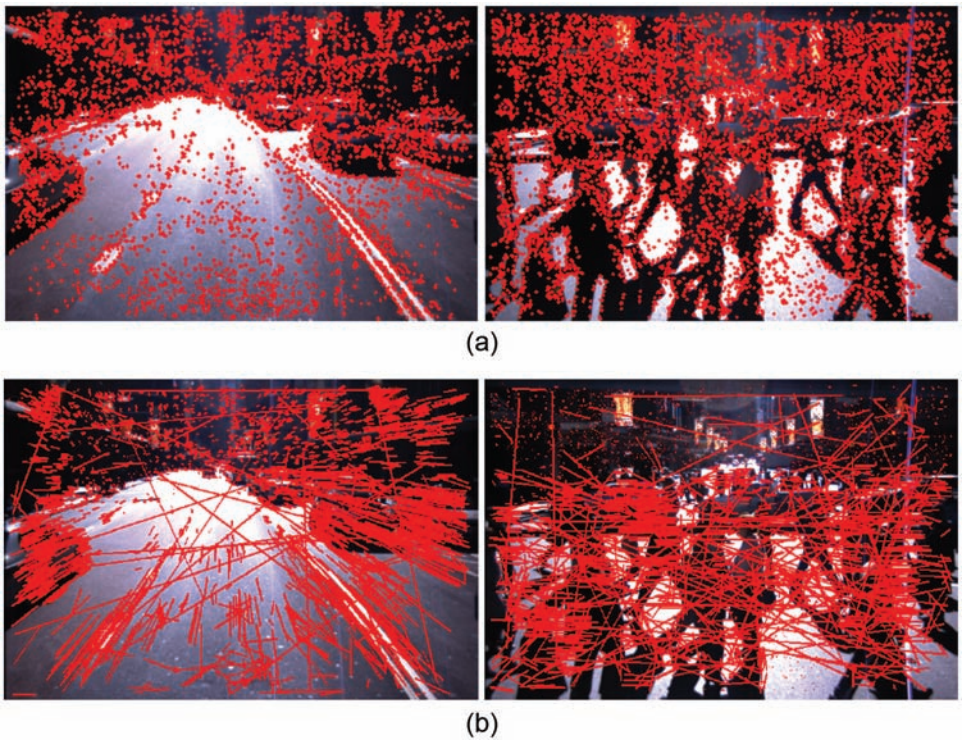

Fig. 2. Examples of the feature (a) extraction and (b) matching processes. In the left column the vehicle is moving forward, whereas in the right column the vehicle is static (all optical flow is generated solely from dynamic objects).

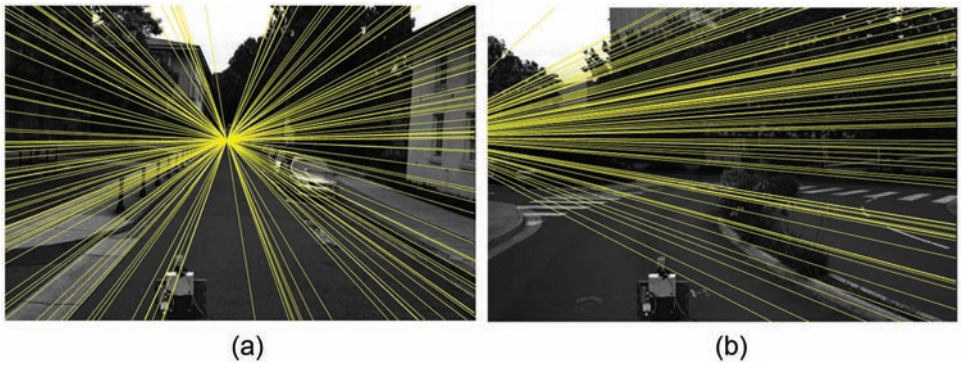

Fig. 3. Examples of epipolar lines for the particular cases of (a) forward motion and (b) counter-clockwise rotation.

Fig. 4. Initial RANSAC classification, based on the image information presented in Figure 2. Each line segment has a color ranging from red (static object) to green (dynamic object), and matching pairs considered outliers were discarded.

between each pair, it does not serve as well in applications that aim at generalization (Lowe, 2004), which is the case in non-parametric modeling based on training data. Because of this, a new descriptor that serves as the input vector $\mathbf{x}_n$ for the GP framework is introduced here and it is of the form

$$\mathbf{x}_n = \{u, v, r, g, b, m\}_n \tag{1}$$

In the equation above, (u, v) are the pixel coordinates of the matched feature in the image (as a convention, the first frame is used for coordinate extraction). This information is necessary in order to correctly model different motion patterns throughout the image (e.g. the top portion of the image is mostly static, whereas the right portion usually contains motion contrary to the vehicle's), and together comprise the spatial component of the descriptor. The four other parameters, $\{r, g, b, m\}$, are the color components and are calculated by placing a w-by-w window centered on (u, v) and extracting the average of pixel intensities in this area (a seven-by-seven window is used during experiments, to minimize the influence of noise). The values of $\{r, g, b\}$ are obtained by applying this method to the red, green and blue channels respectively of the colored image, and m is obtained by applying the same method to its monochromatic version. This color information is necessary in order to correctly model transitions between objects, delineating their borders and allowing the algorithm to 'fill in the gaps' where no features were detected. Other relevant components may be added seamlessly to the descriptor without any further changes in the algorithm.

The final optical flow information set that represents each frame is then of the form $(X, \mathbf{y})_{n=1}^{N}$, where X is a $D \times N$ matrix containing the $D = 6$ descriptor values $\mathbf{x}_n$ for all N matched features and $\mathbf{y}$ is a $1 \times N$ vector containing the probability y_n that each one of these features belongs to a dynamic object. It is important to note that, since this descriptor does not require matching information between features in different frames, it is possible to use it to parametrize any pixel in the image. The descriptor $\mathbf{x}_m$ generated from these pixels will not have a corresponding ground-truth y_m (obtained using RANSAC, which requires matching information), and therefore cannot be incorporated into the non-parametric model, however, it can still be used during the inference stage that provides the final static/dynamic classification. By performing this inference on every pixel, a dense classification of the entire image can be obtained, even though only sparse optical flow information is readily available from each frame pair.

5. GP classification

From a machine learning perspective, the problem of dynamic object segmentation can be seen as a binary classification problem, where each pixel in the image has a certain probability of either belonging to a static or dynamic object. We use here a GP (Rasmussen and Williams, 2006) to perform this classification based on initial ground-truth results obtained according to the RANSAC algorithm (see previous section). Descriptive information from sparse matched features are used to optimize the hyperparameters of a positive-definite kernel (the covariance function) that characterizes the relationship between inputs, and the resulting non-parametric model allows the classification of the entire image as a smooth continuous function, along with the corresponding uncertainty estimates.

5.1. Inference equations and covariance function

A GP is a non-parametric method in the sense that it does not explicitly specify a functional model between inputs and outputs. Instead, it uses information available in a training dataset $\Lambda = (X, \mathbf{y}) = \{\mathbf{x}, y\}_{n=1}^{N}$ to learn the relationship between different points in the input space, and then extrapolates this information to infer the output of new data in a probabilistic manner. A GP model is entirely defined by mean $m(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$ functions:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2}$$

Most implementations assume $m(\mathbf{x}) = 0$ without loss of generality by scaling the data appropriately, and $k(\mathbf{x}, \mathbf{x}')$ is the covariance function, a positive-definite kernel whose coefficients (the hyperparameters) are optimized to maximize a certain objective function. Inference for a single test point $\mathbf{x}_*$ given Λ is calculated as

$$\bar{f}_* = k(\mathbf{x}_*, X)^{\mathrm{T}} [K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y} \tag{3}$$

$$\mathcal{V}(\bar{f}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, X) [K(X, X) + \sigma_n^2 I]^{-1} k(\mathbf{x}_*, X) \tag{4}$$

where σ_n^2 quantifies the noise expected in observation y and K is the covariance matrix, with elements K_{ij} calculated based on the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. Due to the non-stationary behavior of a typical segmented image (abrupt changes from a static to a dynamic object), the neural network covariance function (Williams, 1998) was chosen here to model the relationship between input points. The neural network covariance function is derived from a neural network with a single layer, a bias and $H \to \infty$ hidden units. If the hidden weights are assumed to be Gaussian distributions with zero mean and covariance Σ, this covariance function can be defined (Neal, 1996) as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \arcsin\left(\frac{2 \widetilde{\mathbf{x}}^{\mathrm{T}} \Sigma \widetilde{\mathbf{x}}'}{\sqrt{(1 + 2 \widetilde{\mathbf{x}}^{\mathrm{T}} \Sigma \widetilde{\mathbf{x}})(1 + 2 \widetilde{\mathbf{x}}'^{T} \Sigma \widetilde{\mathbf{x}}')}} \right) \tag{5}$$

where $\widetilde{\mathbf{x}} = (1, x_1, \ldots, x_D)^{\mathrm{T}}$ is an augmentation of $\mathbf{x}$ with the constant value 1 (bias), and σ_f^2 is a signal variance used to scale the correlation between points determined by the neural network covariance matrix Σ (the length-scale matrix, here assumed diagonal with $D+1$ eigenvalues). The expression also contains a sigmoid-like function, $\arcsin(x)$, to model sharp transitions and non-linearities.

5.2. Hyperparameter optimization

During the training stage, the hyperparameters of the covariance function are optimized to minimize a certain objective function, here chosen to be the log-marginal likelihood (Rasmussen and Williams, 2006) due to its ability to balance data fit and model complexity, thus minimizing the chance of over-fitting. The negative log-marginal likelihood objective function is defined as follows:

$$\zeta = \ln p(\mathbf{y}|X) = -\frac{1}{2}\log(|K|) - \frac{1}{2}\mathbf{y}^{\mathrm{T}}K^{-1}\mathbf{y} - N\log(2\pi) \quad (6)$$

The hyperparameter set θ to be optimized is composed of the diagonal elements of Σ (length scales), the signal variance σ_f and the noise level σ_n. Initially, the optimization is conducted using a combination of stochastic maximization (simulated annealing), to avoid local minima, and gradient-descent algorithms to reduce the influence of initial conditions. As new data becomes available during navigation further optimization is necessary, however, since the environment around the vehicle is assumed to change gradually, only a few steps of gradient-descent are performed at each iteration. This gradual optimization also dramatically increases the final computational speed of the algorithm, and empirical data shows that the hyperparameters tend to converge to constant values after a certain period.

5.3. Probabilistic least-squares classification

While the predictive mean $\overline{f}_*$ is useful in determining the most likely hypothesis, it can also be misleading if considered in isolation. One of the key advantages of a GP is its ability to also calculate the variance $\mathcal{V}(\overline{f}_*)$ of each prediction, which acts not only as a way of identifying areas with high measurement uncertainty but also can be combined with the predictive mean to generate a probability distribution that acts as a classifier for the entire input space. We use here a method known as probabilistic least-squares classification (Rasmussen and Williams, 2006), which 'squashes' the predictions through a sigmoid function with parameters α and β determined using a 'leave-one-out' (LLO) approach, for speed purposes. This sigmoid function is introduced in Platt (2000), and the implemented version for training its parameters is defined as

$$p(y_i|X, y_{-i}, \theta) = \Phi\left(\frac{y_i(\alpha\mu_i + \beta)}{1 + \alpha^2\sigma_i^2}\right) \quad (7)$$

where $\Phi(.)$ is the cumulative unit Gaussian, y_{-i} refers to the output values of all training data excluding the pair $(\mathbf{x}_i, y_i)$, μ_i and σ_i are the predictive mean $\overline{f}_*$ and variance $\mathcal{V}(\overline{f}_*)$ at the point $\mathbf{x}_i$, and θ represents the optimized hyperparameters of the covariance function. The training of α and β can be performed by partitioning the original matrix K^{-1} to eliminate the influence of $\mathbf{x}_i$, thus eliminating the need to recalculate the entire covariance matrix for each training point (Wahba, 1990). The new expressions for the LLO predictive mean and variance are presented in equation (8), and they allow the classification of each pixel in the image as a static, dynamic or unsure object, according to user-defined thresholds:

$$\mu_i = y_i - \frac{[K^{-1}y]_i}{[K^{-1}]_{ii}} \qquad \sigma_i^2 = \frac{1}{[K^{-1}]_{ii}} \quad (8)$$

5.4. Incremental updates

One drawback of GPs is the cost of inverting K in equations (3) and (4), in order to respectively calculate the predictive mean and variance. This inversion has a computational complexity of $\mathcal{O}(N^3)$, where N is the number of points, and rapidly becomes the bottleneck in the algorithm's speed as the amount of available data increases. Since ours is an online approach, where the non-parametric model (represented as the covariance matrix K) is generated incrementally during navigation, it is necessary to find a way to both incorporate and remove information from the covariance matrix without having to completely recompute it at every iteration. A common approach in GP literature (Scholkopf and Smola, 2002; Osborne et al., 2008) is using the Cholesky decomposition to calculate the predictive mean $\overline{f}_*$ and variance $\mathcal{V}(\overline{f}_*)$, instead of the covariance matrix K directly. The Cholesky decomposition is useful for solving linear systems with a symmetric, positive-definite coefficient matrix, and produces more numerically stable results than a straightforward matrix inversion. Assuming that the covariance matrix K and the Cholesky matrix C are defined (Smith et al., 2010) as

$$K = \begin{bmatrix} K_{11} & \mathbf{k}_{12} & K_{13} \\ \mathbf{k}_{12}^{\mathrm{T}} & k_{22} & \mathbf{k}_{23} \\ K_{13}^{\mathrm{T}} & \mathbf{k}_{23}^{\mathrm{T}} & K_{33} \end{bmatrix} \quad C = \begin{bmatrix} C_{11} & \mathbf{c}_{12} & C_{13} \\ \mathbf{0} & c_{22} & \mathbf{c}_{23} \\ \mathbf{0} & \mathbf{0} & C_{33} \end{bmatrix} \quad (9)$$

the resulting Cholesky matrix C' obtained by marginalizing (removing) the central row and column is given by

$$C' = \begin{bmatrix} C_{11} & C_{13} \\ \mathbf{0} & \gamma(C_{33}^{\mathrm{T}}C_{33} + \mathbf{c}_{23}^{\mathrm{T}}\mathbf{c}_{23}) \end{bmatrix} \quad (10)$$

where γ is the Cholesky update operator, readily available in packages such as (Dongara et al., 1979; MATLAB, 2011) and which exploits the special structure of $\mathbf{c}_{23}^{\mathrm{T}}\mathbf{c}_{23}$ to attain a computational complexity of $\mathcal{O}(n^2)$. The marginalization of K is obtained simply by removing its middle row and column. Similarly, if the covariance matrix K and the Cholesky matrix C are defined as

$$K = \begin{bmatrix} K_{11} & K_{13} \\ K_{13}^{\mathrm{T}} & K_{33} \end{bmatrix} \quad C = \begin{bmatrix} C_{11} & C_{13}^{\mathrm{T}} \\ \mathbf{0} & C_{33} \end{bmatrix} \quad (11)$$

the resulting Cholesky matrix C' obtained by expanding (adding) a central row and column is given by

$$C' = \begin{bmatrix} C_{11} & C_{11}^{\mathrm{T}} \backslash k_{12} & C_{13} \\ \mathbf{0} & \sqrt{k_{22} - \mathbf{c}_{12}^{\mathrm{T}} \mathbf{c}_{12}} & \frac{\mathbf{k}_{23} - \mathbf{c}_{12}^{\mathrm{T}} C_{13}}{c_{22}} \\ \mathbf{0} & \mathbf{0} & \gamma(\, C_{33}^{\mathrm{T}} C_{33} + \mathbf{c}_{23}^{\mathrm{T}} \mathbf{c}_{23}) \end{bmatrix} \tag{12}$$

where the operator $\backslash$ is used to indicate the solution of $\mathbf{c}_{12}$ in the equation $C_{11}^{\mathrm{T}} \mathbf{c}_{12} = \mathbf{k}_{12}$, obtained by the use of backwards or forward substitution for the upper triangular matrix C_{11}. Again, the expansion of K is done directly by incorporating the middle row $\mathbf{k}_* = [\mathbf{k}_{12}^{\mathrm{T}}, k_{22}, \mathbf{k}_{23}]$ and column $\mathbf{k}_*^{\mathrm{T}}$, obtained by calculating the covariance between $\mathbf{x}_*$ and all training points $\mathbf{x}_n$ (points before the middle are part of $\mathbf{k}_{12}$, points after the middle are part of $\mathbf{k}_{23}$, and k_{22} is the diagonal term of the new row and column).

5.5. *Information selection*

The feature extraction and matching techniques described in Section 4 produce an average of around 7000 new features per frame. It is impractical to incorporate all these new data points into the GP framework at every iteration, as it would create an exceedingly complex and computationally prohibitive model after just a few iterations. Fortunately, most of this information is redundant, since it describes an environment that is only gradually changing due to camera motion and the presence of dynamic objects, and therefore can be safely discarded without compromising model accuracy. Also, it is necessary to remove erroneous information generated by RANSAC misclassifications, that could skew results and compromise long-term accuracy. In order to determine which data points should be incorporated and which should be removed from the GP framework at each iteration, we impose a series of four filtering steps over different subsets of the available data, designed to remove redundant and unreliable information from the non-parametric model. Each of these steps is described below in more detail:

1. **Density constraint on input data.** As stated previously, an average of 7000 new data points are produced at every iteration by the feature extracting and matching techniques. Fortunately, most of this information is redundant, since features tend to be clustered into specific portions of the image and may share a similar classification. This allows them to be safely discarded without compromising results by performing a nearest-neighbor search in the input space for each feature and discarding those whose classification is similar to their closest neighbors, according to a certain distance threshold. This process is repeated until only one feature of any given class remains in each portion of the input space determined by the distance threshold, and if any portion is represented by both classes then two features are maintained, one for each class.
2. **Inference on remaining input data.** The next step consists of performing inference on the remaining input data points, which were not discarded by the density constraint. This inference process provides a final GP classification for these data points based on the current non-parametric model (this step is skipped in the first iteration), which is then compared to the initial RANSAC classification. Data points that are *correctly* classified (the GP classification is the same as the RANSAC classification) are discarded, because their position in the input space is already well-represented and does not require more information to provide accurate estimates. Those that were *incorrectly* classified are assumed to be relevant and are incorporated into the non-parametric model, increasing the amount of information available for inference.
3. **Inference on the non-parametric model.** The next step is to perform inference on the non-parametric model itself. This time, data points that are *incorrectly* classified are removed, thus decreasing the amount of information available for inference. This step is important in eliminating RANSAC misclassification (outliers), as they are assumed to be a minority and therefore less representative of their position in the input space. The removal of such misclassifications allows the non-parametric model to maintain consistency even after long periods of navigation, essentially forgetting old environment behaviors and adapting to new ones.
4. **Density constraint on the non-parametric model.** Lastly, the density constraint is enforced on the non-parametric model itself, removing data points with similar classification that are close to each other according to a certain distance threshold. As in the first step, this serves as a way to decrease the amount of redundant information available for the non-parametric model while still maintaining its spatial distribution in the input space.

Figure 6 shows the progression of the size of the non-parametric model during navigation. It starts empty, without any information, and at the first iterations a large number of data points are incorporated, because the algorithm is still learning the different static and dynamic structures of the environment. When the number of incorporated data points reaches around 4000 the size of the non-parametric model stabilizes, with roughly the same number of data points being incorporated and removed at each iteration. This is to be expected, since as a general rule the environment changes gradually with each frame, and the algorithm is capable of learning new behaviors at the rate at which it is forgetting old ones. When there is a sudden change in the environment (e.g. the camera starts/stops moving, or a previously dynamic object becomes static or vice versa) there is a spike on the number of data points incorporated, indicating that the environment has suddenly become more complex and the algorithm is trying to learn this new configuration. Once it has managed to do so, the size of the

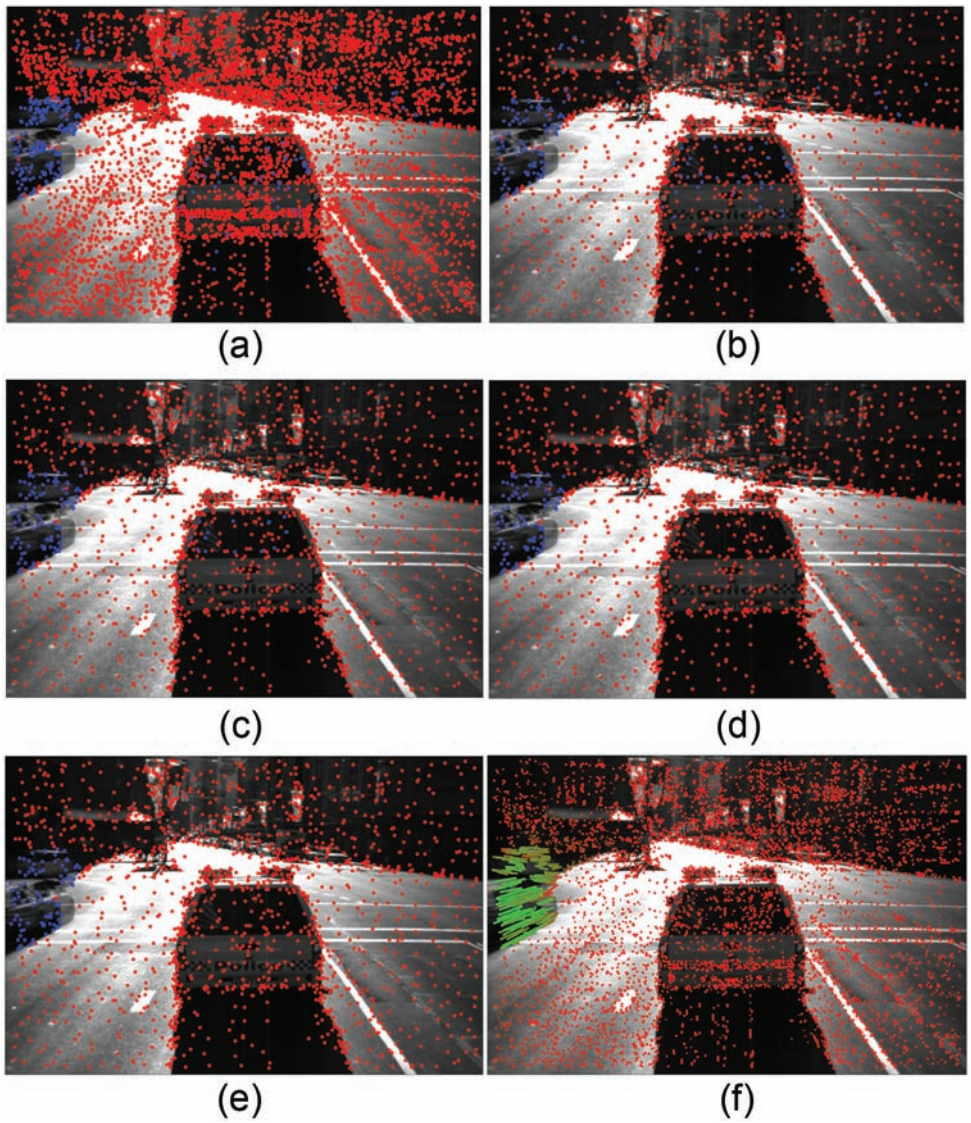

Fig. 5. Stages of information selection (red dots are features classified as static, and blue dots are features classified as dynamic). (a) Initial input data points. (b) Input data points after density constraint. (c) Data points after inference that are incorporated into the non-parametric model. (d) Non-parametric model data points after inference. (e) Non-parametric model after density constraint. (f) Final GP classification of all input points, ranging from red (static) to green (dynamic).

non-parametric model stabilizes and returns to the 4000 level, which is maintained even after an undetermined long period of navigation. This level can be adjusted by fine-tuning the value for the distance threshold in the density constraint, according to requirements in performance and computational efficiency.

5.6. Pixel-by-pixel classification

In principle, the entire image can be classified using the probabilistic least-squares classification technique described in Section 5.3, by calculating the input vector for each pixel in the image and performing GP inference based on the current non-parametric model. However, inference has a computational complexity of $\mathcal{O}(n^2)$, with n being the number of data points available, and for a 600×400 image a total of 240,000 inferences would have to be performed. Needless to say, these numbers make a dense classification of the entire image infeasible for real-time deployment.

Since each inference is performed independently, sharing the same covariance matrix K and ground-truth vector $\mathbf{y}$, one possible solution would be to parallelize the inference process, or to perform the matrix multiplications in a graphics processing unit (GPU). Another solution proposed here is pixel subsampling, in which only certain pixels are selected for inference and the remaining ones are calculated based on their neighbors' properties. For this particular application, a predetermined number of rows and columns are skipped during inference (e.g. every other row and column is skipped), and their classification is calculated

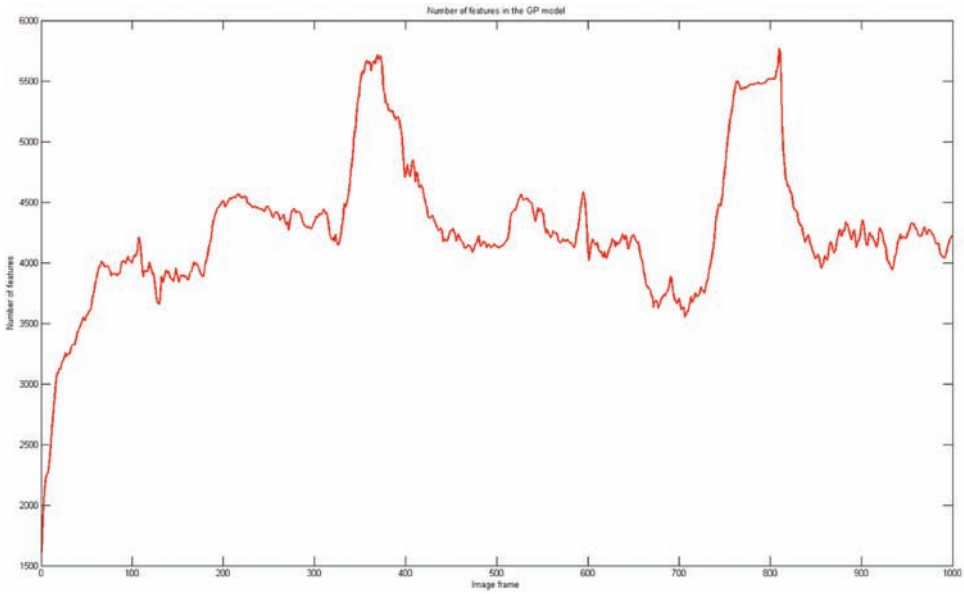

Fig. 6. Number of data points maintained by the non-parametric model at each iteration. Note that this number stabilizes at around 4000, with spikes indicating moments in which there was a radical change in the environment (e.g. the camera started/stopped moving, or a static object suddenly became dynamic or vice versa). As the algorithm learns these new behaviors, the number of data points stabilizes and returns to the 4000 level.

using linear interpolation based on their nearest neighbors both horizontally and vertically. This approach decreases the computational complexity by a factor of r^2, where r is the number of rows (and columns) skipped, and by fine-tuning this parameter it is possible to achieve the desired computational complexity. It also has the added benefit of blurring the borders of dynamic objects, thus providing a 'safety zone' that minimizes the chances of using false information for visual odometry and filling in the gaps in small patches where no texture is available.

An example of such interpolation is shown in Figure 7, in which each pixel in the image is depicted by its probability of belonging to a dynamic object. In Figure 7(a), only every fifth pixel in each row and column is classified, providing an outline of the final classification that is then completed via interpolation as shown in Figure 7(b). The resulting probabilistic classification can then be transformed into a discrete classification between static/dynamic/unsure objects, based on user-defined thresholds and the uncertainty estimates for each pixel.

6. Experimental results

This section presents and discusses the experimental results obtained using the proposed algorithm for the self-supervised segmentation of dynamic objects. Initially, data collected from a modified vehicle navigating in an urban environment is used to generate a dense pixel-by-pixel classification, which serves to qualitatively validate the proposed algorithm. These results are then compared with traditional approaches for the segmentation of dynamic objects, as a way to provide a quantitative measurement of their accuracy in relation to well-established similar techniques. The same algorithm is also tested, without any modification, using another dataset collected from a portable camera device. This serves to show its ability to generalize over different camera configurations, environments and vehicle dynamics.

Afterwards, visual odometry results obtained using the semi-parametric coupled Gaussian processes (SPCGP) framework (Guizilini and Ramos, 2013) are presented both with and without the incorporation of the proposed algorithm. These results testify to its ability in correctly segmenting dynamic objects, which can then be removed to allow the use of a purely static background for visual odometry purposes. Finally, a technique concerning the clustering of dynamic pixels into different object instances is introduced and discussed, along with results obtained using the final GP segmentation from the urban dataset. It is then shown how to further cluster these objects into different categories, which contain semantically meaningful information regarding their members even though no prior knowledge of the environment is taken into consideration.

6.1. Dynamic object segmentation

The segmentation proposed algorithm was tested in two different circumstances: an urban environment, using data collected with a modified vehicle equipped with a single uncalibrated camera, and a pedestrian perspective, using data collected with a portable camera device. The urban environment results were then compared with other approaches to dynamic object segmentation, as a way to validate how our algorithm improves over already existing

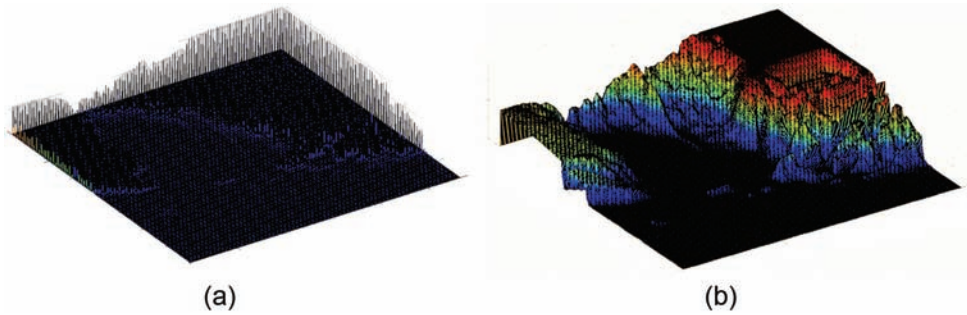

Fig. 7. Example of pixel subsampling. In (a), only every fifth pixel in each row and column is classified, providing an outline for the final classification that is then completed in (b) using linear interpolation.

techniques. The portable camera device results, on the other hand, testify to our algorithm's ability to generalize over different environments and camera configurations without any changes in parameters.

6.1.1. Urban scenario. The urban dataset is composed of 14,500 images obtained at a rate of three frames per second along a trajectory of roughly 10 km, in which the vehicle interacted normally with pedestrians, cars and buses at speeds of up to 50 km/h. During this trajectory the images collected were subject to radical changes in luminosity, due to cloud coverage and tall buildings, and also to a wide variation in structures and dynamic objects around the vehicle, as it navigated through different portions of the city.

The algorithm was initialized without any training, using random hyperparameters and an empty covariance matrix. At each iteration a new set of features is selected and incorporated into the non-parametric model, and redundant features are removed in order to keep the computational cost within a certain boundary. The remaining data points are used for training, initially using stochastic optimization to avoid local minima and afterwards gradient-descent techniques to deal with gradual changes in the environment structure. The first few iterations of the algorithm were done in less than a second per frame, however, as new information was incorporated the computational time stabilized at a few seconds per frame, mostly due to the frequent inferences necessary to decide which data points should be added/removed from the non-parametric model. This time estimate was obtained in a Matlab implementation, using a single CPU and no GPU acceleration. We are confident that a proper optimization could generate great improvements in speed, due to a high parallelization potential. Approximation techniques, such as sparse GPs (Csat and Opper, 2002; Rasmussen and Williams, 2006), could also substantially improve computational speed, since it is natural to assume that distant features should have no impact in calculations.

The final GP segmentation results obtained based on the initial RANSAC classification introduced in Figure 5 are depicted in Figure 8. A dense classification was performed, with inference being conducted using every fifth pixel in each row and column to provide an outline of the entire image, and the remaining pixels were classified using linear interpolation for speed purposes. This process was done for both the predictive mean (Figure 8(b)) and variance (Figure 8(c)) values, which indicate, respectively, the best hypothesis for each probability distribution and the confidence with regard to that hypothesis.

In Figure 8(b) we can see that the algorithm was indeed able to correctly detect most of the dynamic objects in the environment, segmenting them from the static background. Virtually all misclassifications given by the initial RANSAC classification in Figure 4(a) were removed, especially on the top left and right corners of the image, and also in the areas where the street was represented. We attribute this to the selection process, which is capable of removing outliers and detecting the correct tendency of each portion of the image even in the presence of significant noise. Also, virtually all the featureless regions in Figure 4(b) were correctly filled by the dense classification process provided by the GP framework, allowing the complete delineation of all dynamic objects and their boundaries in relation to the static background. Empirical tests suggest that 1.5 s (five frames) is enough time for the proposed algorithm to learn the behavior of an object in the environment, be it a new dynamic object or a static object that suddenly became dynamic (or vice versa). Note that this response time decreases if the behavior in question has already been observed previously (e.g. new dynamic objects appearing where they usually do), which is the case most of the time.

Another key benefit of using the GP framework for segmentation is its ability to calculate the uncertainty inherent in each estimate, thus providing a measurement of variance for each pixel alongside the predictive mean. In the context of object detection, this variance can be used to determine which portions of the image are most likely to be correctly classified and which require more information before a final classification can be made, forming the basis for active learning (Dima, 2006; Settles, 2010). Examples of such variance are shown in Figure 8(c), with darker areas representing lower uncertainty values and lighter areas representing higher uncertainty values. It is clear that most of the variance is concentrated in the borders of the image, which is to be expected since this region is where the feature

Fig. 8. Examples of the final GP classification results, based on the initial RANSAC information depicted in Figure 4. (a) Original images. (b) Predictive mean, defined by a number ranging from 0 (black, static object) to 1 (white, dynamic object). (c) Predictive variance, normalized to values ranging from 0 (lowest) to 1 (highest).

density is lowest (lots of features disappear and appear between frames due to vehicle motion) and also where new objects are detected for the first time. As we move to the central portions of the image new objects and features are gradually incorporated into the non-parametric model and the variance decreases.

More examples of the predictive mean obtained using the proposed algorithm are shown in Figure 9. (A video showing the evolution of the predictive mean and variance in more detail during navigation can be found in Extension 1.) These results were obtained in the same run, meaning that the non-parametric model, starting empty and with random hyperparameters, had to constantly adapt to changes in the environment in order to learn the characteristics of each individual frame. The information selection process described in Section 5.5 was used to keep computational complexity manageable during navigation, and the number of data points maintained in each iteration for the first 1000 frames is depicted in Figure 6. From these images note that even though the vehicle experienced radical changes in both local and global luminosity, environment structures and a wide range of different dynamic objects, it was still capable of providing accurate segmentation results.

The proposed algorithm for the segmentation of dynamic objects was compared with other approaches to dynamic object segmentation, and the results are presented in Figure 10. These results were obtained based on information from 200 hand-labeled images randomly selected from the 14,500 images available for testing. The dotted line represents the initial RANSAC classification results, the black line indicates the proposed algorithm, and the red line indicates the proposed algorithm but with a square-exponential

Fig. 9. Final segmentation results. Each pixel is defined by a number ranging from 0 (black, static object) to 1 (white, dynamic object). These results were obtained in the same run as the non-parametric model constantly adapts to new environment characteristics, without any prior information and without any human intervention.

covariance function, instead of the neural network covariance function. The blue line indicates results obtained using SVMs (Cristianini and Shawe-Taylor, 2000) as the self-supervised classification method instead of GPs, and the green line indicates the optical flow classification (OFC) results obtained based on Namdev et al. (2012).

The OFC uses motion potentials based on geometry to build a graph-like structure from dense optical flow and feature tracking (the SLAM component was not implemented here, as it can be equally applied to any methodology). This graph is then clustered together and nodes with similar potentials become motion segments that generate a single structure. The receiver operating characteristics (ROC) curves for each of these approaches are shown in Figure 10(a), where it is possible to see that the proposed algorithm outperforms the others in all threshold levels, and in particular that it improves over the initial RANSAC classification by a significant margin. It is also possible to see the importance of covariance function selection, since the same algorithm performed significantly worse when the square-exponential covariance function was used.

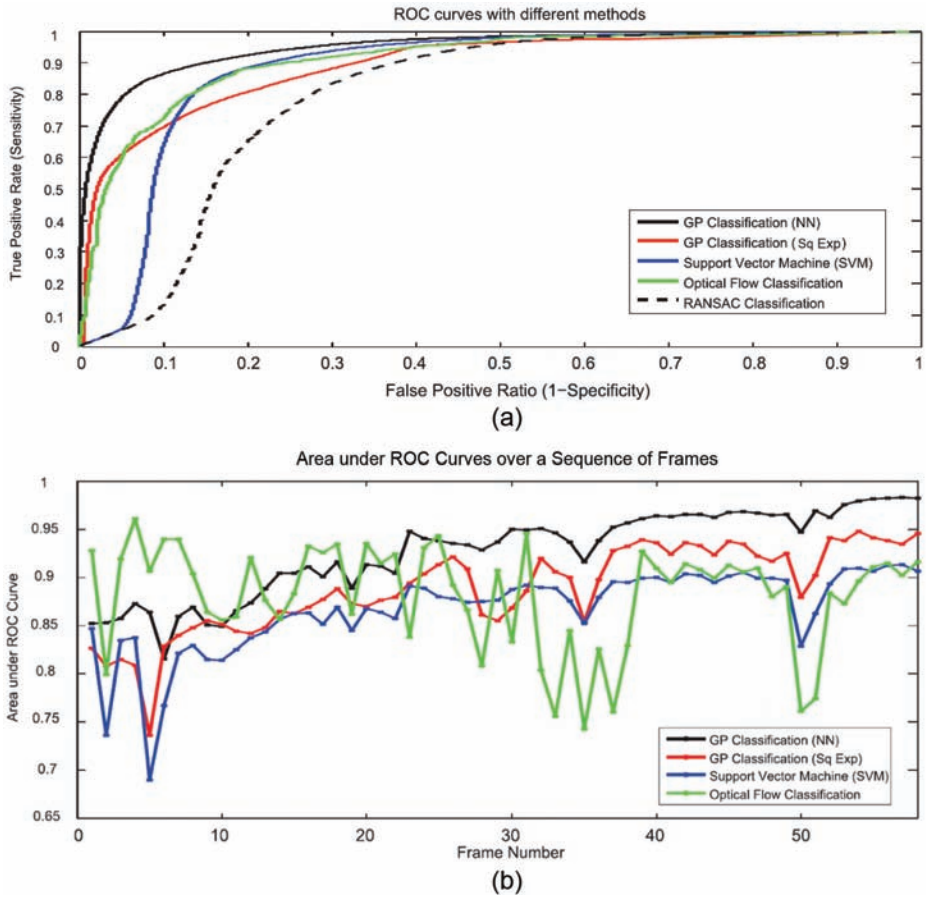

Fig. 10. Comparison of the proposed algorithm for the segmentation of dynamic objects with other similar approaches. (a) ROC curves for each approach. (b) Area under the ROC curves for each approach in different frames. NN: neural network kernel; Sq Exp: squared exponential kernel.

Figure 10(b) depicts the area under the ROC curve for each subsequent frame at the beginning of navigation, indicating how accuracy changes as new data is incorporated into the non-parametric model. As expected, the OFC approach does not improve over time, since it is not based on learning techniques, using instead individual information from each frame. The accuracies of the three other approaches increase steadily over time, with occasional drops that indicate moments in which there was a significant change in the environment (e.g. the camera started/stopped moving, or a new object entered the camera's field of vision), and the proposed algorithm consistently outperforms the other two. It is interesting to note that, at the beginning of navigation, the OFC is the best solution, since there has been no time for the non-parametric model to learn the environment characteristics. However, after a few frames the proposed algorithm improves and becomes the best solution, while the OFC oscillates heavily at each iteration.

As a purely visual-based algorithm, the proposed framework is highly sensitive to the quality of images provided by the camera system. A histogram filter was applied to account for global luminosity changes, however, local shadow, especially from tall buildings nearby, could trick the algorithm into thinking that large portions of the environment were dynamic. Shadow removal techniques (Finlayson et al., 2002) could in principle eliminate this issue, and also increase the number of static features available for correct model estimation by RANSAC. Other challenges, such as heavy rain, darkness or severe occlusion, could also compromise performance, as they would with any other vision-dependent approach. As a machine-learning-based algorithm, the proposed framework is able to adapt to gradual changes in the environment, increasing response time and performance with previously modeled objects and learning the behavior of new ones as they are introduced. However, this can lead to the learning of wrong patterns (e.g. if most of the field of view is covered by a dynamic object, such as a truck), which could also compromise performance. Once these false patterns are removed, though, the algorithm should be able to recover and forget this model in favor of the correct one.

6.1.2. Portable camera device. The same algorithm, without any further modifications, was also tested with images obtained using a portable camera device (Figure 11(a)),

as a way to qualitatively explore its ability to generalize over different configurations and environments. The non-parametric model was initialized empty from random hyperparameters, and the shakiness of the camera posed a challenge to the RANSAC algorithm, since now the baseline between frames was much smaller and its motion unconstrained by 2D vehicle dynamics. The results obtained using this configuration are presented in Figure 11(b), where it is possible to see that again there is a wide variation in luminosity and structures encountered during navigation, and the algorithm was still capable of correctly segmenting most of the dynamic objects in each scene.

Still, it is possible to see some shortcomings in these results, mostly due to the presence of shadows, which are classified as dynamic objects because of the purely visual nature of the segmentation technique. Also, far-away dynamic objects are usually undetected, due to low relative speeds that are dismissed as noise. Sudden changes in texture and color tend to generate gaps in the segmentation results, because the high-dimensional input vector used by the GP relies both on spatial coordinates and color intensity information. However, it is worth emphasizing that these results were obtained without any human intervention whatsoever, based solely on non-labeled image information collected iteratively from a single uncalibrated camera during navigation.

6.2. Improvements in visual odometry

We present here one possible application of the proposed algorithm for the self-supervised segmentation of dynamic objects, in the field of autonomous navigation. Visual odometry, in its standard derivation, requires a static environment in order to provide motion estimates between frames, since in this particular scenario all optical flow generated by features will be solely due to the camera's own translation and rotation. If dynamic objects are present, there will be a component of error in optical flow information that can skew results, especially in an incremental implementation of visual odometry. Unfortunately, most real environments will contain dynamic structures (or at least the possibility of dynamic structures), so it is necessary to deal with this error component in optical flow information before visual odometry can truly provide accurate localization estimates.

The most intuitive solution to this problem is simply to remove the dynamic portion of the environment, leaving only the static background to be used during visual odometry calculations, and this is exactly what the proposed segmentation algorithm is capable of doing. An example of the influence of dynamic objects in visual odometry is presented in Figure 12(a), where the SPCGP framework (Guizilini and Ramos, 2013) was used in the aforementioned urban dataset to provide an estimate of the trajectory navigated by the vehicle. In the visual odometry estimates (red line) it is possible to see several sharp turns, caused mostly by the presence of dynamic objects when the vehicle was not moving (e.g. at a traffic light, with vehicles crossing its path) that tricked the system into thinking the vehicle was rotating. There is also a systematic drift caused by a larger optical flow in one portion of the image (the opposite lane, where vehicles were moving towards the camera with a higher relative velocity), that skews results and over time compromises localization estimates.

Localization results obtained after incorporating the proposed segmentation algorithm, using the same dataset but discarding the dynamic features in each frame, are depicted in Figure 12(b). As expected, there is still some residual drift caused by the incremental nature of visual odometry estimates, however, virtually all sharp turns are removed, allowing the system to recover the overall trajectory shape in great detail. It is worth noting that no extra information was added, since the same dataset used for visual odometry was also used for dynamic object segmentation, without any manual labeling or human intervention whatsoever. The same results, now using the full SPCGP framework (with online updates and a semi-parametric model, as described by Guizilini, 2013), are presented in Figure 13.

6.3. Dynamic object clustering

The segmentation algorithm, as described in the previous section, does not make any distinctions in regard to different dynamic objects. Each pixel is probabilistically classified between the static and dynamic classes, based solely on the current non-parametric model, and each classification is individual. However, by exploiting optical flow patterns and spatial coordinates/colour information from different matched features it is possible to determine the boundaries from each specific dynamic object. It is natural to assume that a (non-deformable) object would have features that share a similar optical flow distribution, and that this distribution changes gradually during navigation. This would allow the tracking of dynamic objects over time, as a way to increase robustness and decrease ambiguity in object segmentation.

This clustering of dynamic pixels is conducted according to the following iterative process, which takes place after the final GP classification:

1. A random feature is selected on the image, forming the core of a new dynamic cluster.
2. All its neighbors within a certain radius are checked. Features with an optical flow pattern whose magnitude/orientation are similar within a certain threshold are added as part of the same dynamic cluster.
3. Step 2 is repeated for all the newly added features, gradually increasing the size of the current dynamic cluster.
4. When there are no new features to be added, the process stops and the current dynamic cluster is determined as a new object instance. Step 1 is repeated for a new random feature that still does not belong to any cluster.

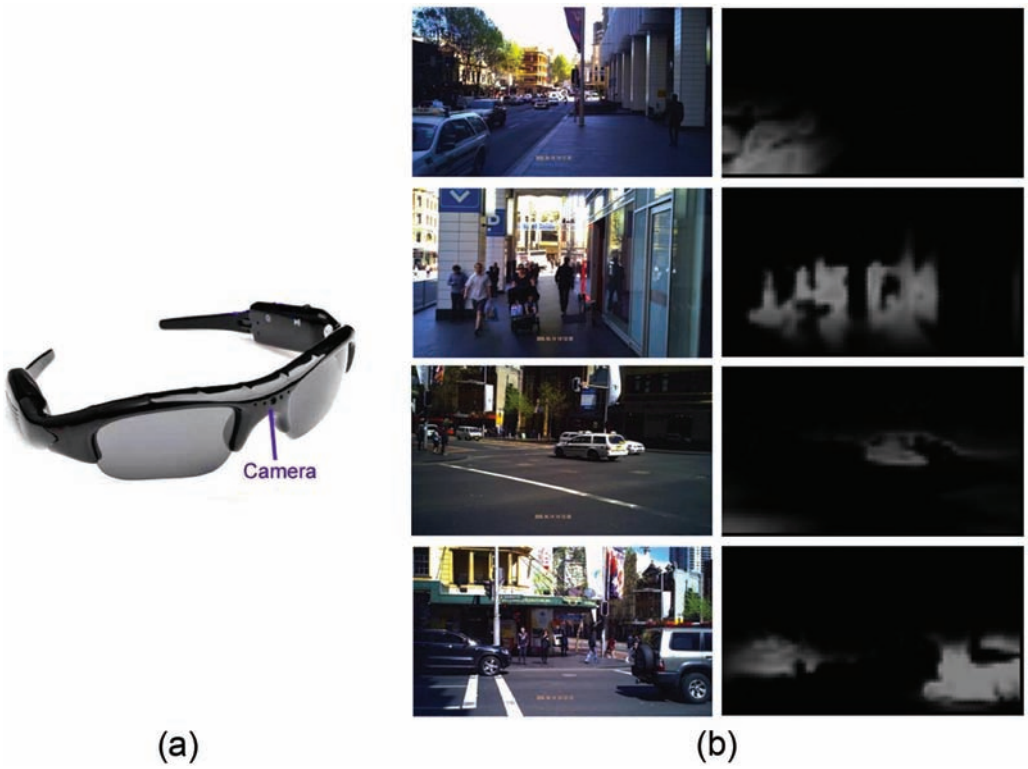

Fig. 11. Dynamic object segmentation results in different frames using a portable camera device.

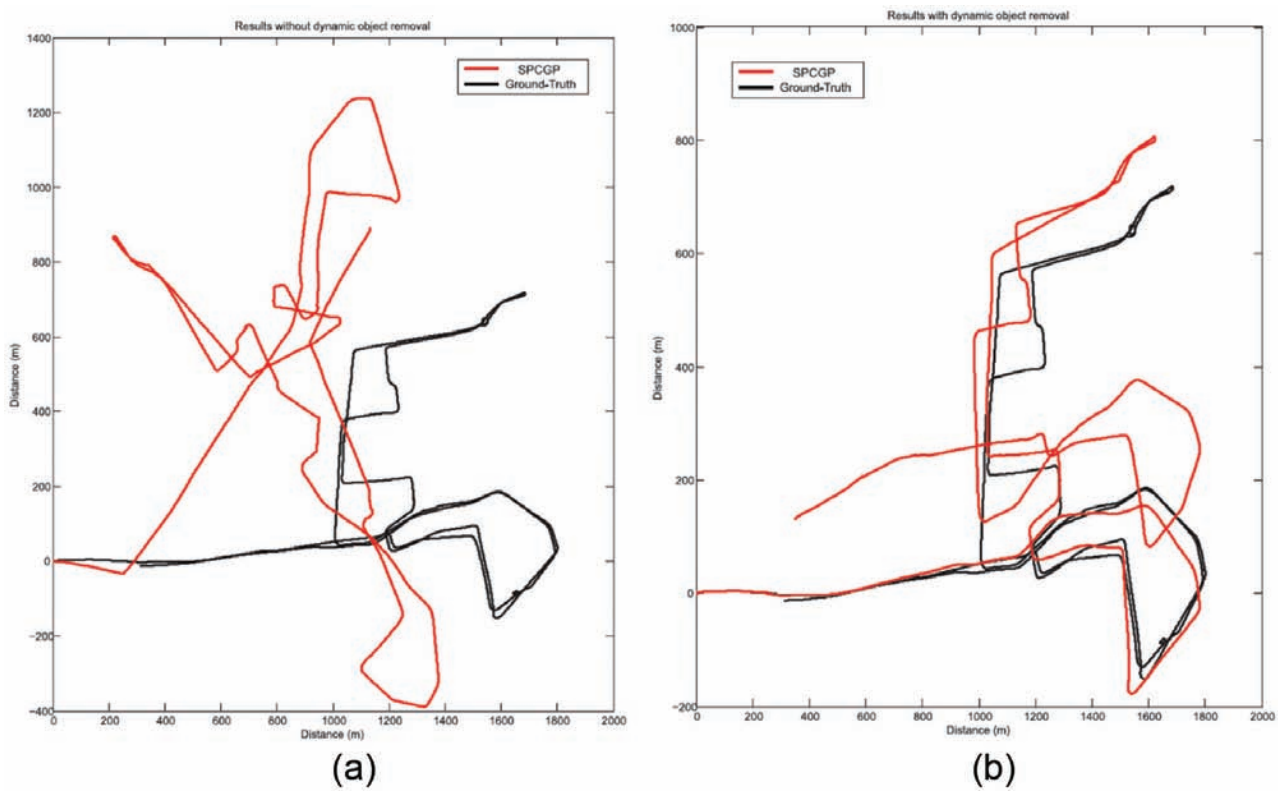

Fig. 12. Localization results in a highly dynamic environment using the SPCGP approach. (a) Without dynamic object removal. (b) With dynamic object removal.

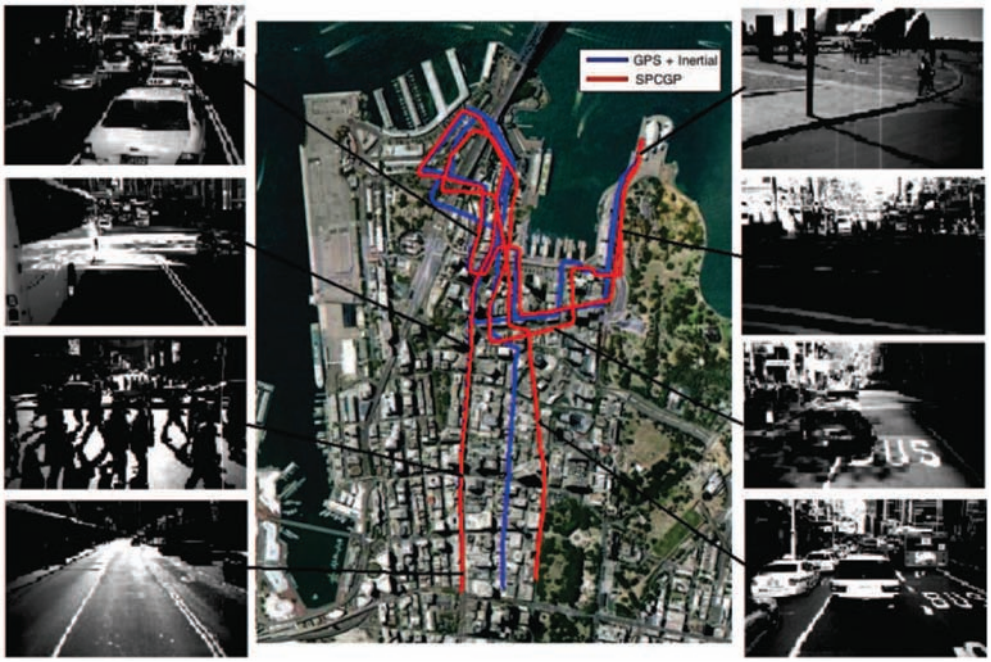

Fig. 13. Visual odometry results using dynamic object removal and the SPCGP framework (Guizilini, 2013).

5. When all features already belong to a cluster, a merging process is conducted to join clusters whose magnitude/orientation average are similar within a certain threshold, and their features share an overlapping area in the image. This step is important to reduce the number of sub-clusters of a single object, due to small variations in color and optical flow distribution.
6. Once the merging process is done, filtering is conducted to remove clusters with a number of features that fall below a certain threshold. This step is important as it removes small clusters generated by noise in the segmentation algorithm.
7. Each cluster is expanded to include pixels that do not contain matched features with optical flow information, according to the dense classification created by the GP framework. Each feature in each cluster is expanded to include its neighboring dynamic pixels, in a process that is repeated until there are no more pixels to expand to. If the same pixel is neighbor to two different clusters, a linear combination of color information and Euclidean distance to the nearest matched feature is used to determine which cluster it should be incorporated in.

Once all dynamic objects are obtained (Figure 14), the next step is to further divide them into different categories, according to the object each one represents (in an urban environment, mostly cars and pedestrians). This is done using data collected from 1000 consecutive images, all segmented according to the algorithm described previously, for a total of 2077 objects. Each object is described using a histogram for each one of its color components (red, green and blue), divided into six bins. These histograms are normalized to have an unitary sum, to account for objects of different sizes. These descriptors are then categorized using *k*-means (Kanungo et al., 2002), a clustering method which aims to partition *n* observations into *k* clusters, where each observation belongs to the cluster with the nearest mean. For the experiments presented here, we selected 8 as the number of clusters to be formed, as a way to minimize the impact of random objects that do not fall within any other category.

Figure 15 shows the results for the three clusters with the most samples, obtained using the method described above, along with some examples belonging to the other five clusters. It is important to note that these categories do not imply any knowledge of the environment, and were generated based solely on the dynamic objects collected during navigation. Nevertheless, we can see a clear pattern present in each one of them, indicating that *k*-means was able to correctly cluster these dynamic objects into semantically meaningful categories. Figure 15(a) contains mostly pedestrians, Figure 15(b) contains mostly cars and other vehicles, and Figure 15(c) contains mostly shadows from pedestrians. Other clusters include mostly partial objects: ones which were not merged into a single group for some reason, shadows from static objects, ones which were classified as dynamic due to the learning nature of the self-supervised algorithm, and other sporadic misclassifications. Tests were conducted using different numbers of clusters, and while a higher number did not show any significant improvement, smaller numbers show a merging between different clusters that affected results.

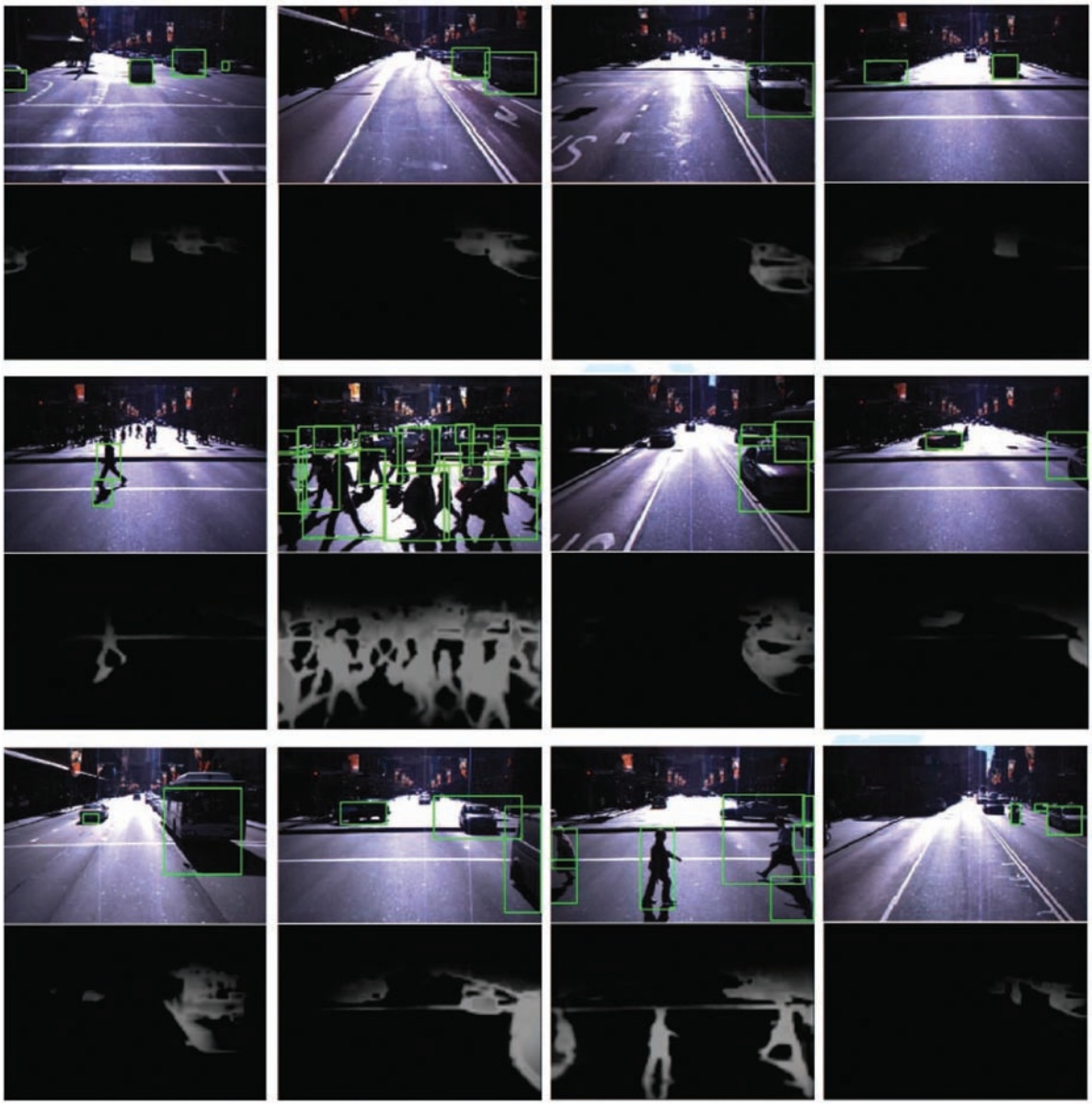

Fig. 14. Different object instances obtained during the iterative clustering process.

To provide quantitative evaluation, a ground-truth dataset was generated by manually labeling 250 dynamic objects as being in one of these three main categories and a fourth one, comprising the remaining five clusters. Several different clustering algorithms were tested (see Table 1) based on both accuracy percentage and V-measure, as described by Rosenberg and Hirschberg (2007). The accuracy percentage values were calculated as the ratio of correctly labeled objects to the size of the entire ground-truth dataset, and V-measure values are proportional to the desirable clustering properties *h* (homogeneity) and *c* (completeness). From these results we can see that *k*-means outperforms other similar techniques, with an accuracy of roughly 71% to 28% obtained with random clustering. A confusion matrix showing the *k*-means results is presented in Table 2, where we can see how the clustering errors are distributed in different categories. As expected, pedestrians and shadows are more easily mistaken for each other, whereas pedestrians and cars have a small overlap. Also as expected, the five remaining categories (labeled as *others*) share a similar error overlap with the other three categories, indicating that they do not possess any specific characteristic that could be semantically meaningful.

7. Conclusion and future work

This paper presented a novel approach to dynamic object segmentation using a single mobile uncalibrated camera, based on self-supervised learning and without human intervention of any sort. Sparse matched features were sorted

Fig. 15. Clustering results for different dynamic objects. (a) Pedestrians. (b) Vehicles. (c) Pedestrian shadows. (d) Examples of objects that do not belong to any of the three main categories.

Table 1. Performance values for clustering with different techniques.

Clustering Method	Accuracy (%)	V-measure $\frac{2hc}{(h+c)}$
k-means (Hartigan and Wong, 1979)	71.3	0.46
Fuzzy *c*-means (Bezdek, 1981)	65.8	0.31
Shadows (Gupta and Chen, 2011)	59.4	0.29
Hierarchical (Gibbons and Chakraborti, 2003)	55.7	0.24
Random	28.2	0.07

Table 2. Confusion matrix for the clustering results.

	Cars	Pedestrians	Shadows	Others
Cars	58	3	2	8
Pedestrians	4	43	11	6
Shadows	4	7	21	8
Others	2	3	5	15

according to the RANSAC algorithm to generate an initial classification between a static background and dynamic foreground objects. This initial classification was then refined using a GP, a non-parametric Bayesian inference

technique, in an online manner. Experiments conducted in an urban scenario, with data collected from a vehicle navigating at speeds of up to 50 km/h while interacting with pedestrians, cars and buses, show promising results that outperform other similar segmentation algorithms and influence visual odometry localization estimates positively. Tests conducted with a portable camera device testify to the proposed algorithm's ability to generalize to different environments and visual systems, and. Lastly, a technique for the clustering of dynamic pixels into different object instances, and the further clustering of these object instances into semantically meaningful categories, was introduced and discussed.

Since it relies purely on visual information, the proposed algorithm also segments shadows as part of dynamic objects, which could be readily addressed by incorporating shadow-removal techniques into the framework. Also, the algorithm's ability to segment far-away dynamic objects is limited because of their small relative sizes and low relative speeds, which poses a challenge to feature extraction and increases the chances of an initial RANSAC misclassification. The incorporation of object-tracking techniques could address this problem, where the algorithm follows each specific object over a sequence of frames and learns its visual information at different scales and viewpoints. The clustering of dynamic pixels into distinct objects and categories allows for the self-supervised creation of a library of dynamic objects, extending the initial binary classification to a more semantically meaningful territory. This dynamic object library can be used to improve the segmentation algorithm itself, creating the possibility of a 'potentially dynamic' class, where a certain object is inherently dynamic but is currently stationary (e.g. a parked car). Future work will focus on algorithm speed, exploring assumptions such as sparsity to decrease computational costs and enable real-time model updating and inference.

Funding

This research was supported under Australian Research Council's Discovery Early Career Research Awards (project number DE120103051).

References

Almanza-Ojeda DL and Ibarra-Manzano MA (2011) 3D visual information for dynamic objects detection and tracking during mobile robot navigation. In: Topalov AV (ed) *Recent Advances in Mobile Robotics*. InTech, pp. 1–23.

Arras K, Mozos M and Burgard W (2007) Using boosted features for the detection of people in 2D range data. In: *International conference on robotics and automation*, pp. 3204–3407.

Bak A, Bouchafa S and Aubert D (2014) Dynamic objects detection through visual odometry and stereo-vision: a study of inaccuracy and improvement sources. *Machine Vision and Applications* 25(3): 681–697.

Bay H, Tuytelaars T and Gool LV (2006) Surf: Speeded up robust features. In: *ECCV*, pp. 404–417.

Bezdek JC (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA: Kluwer Academic Publishers.

Bishop CM (2006) *Pattern Recognition and Machine Learning* (*Information Science and Statistics*). New York, NY: Springer-Verlag.

Chapelle O, Schölkopf B and Zien A (2006) *Semi-Supervised Learning*. Cambridge, MA: The MIT Press.

Cheng G and Zelinsky A (1998) Goal-oriented behaviour-based visual navigation. In: *International conference on robotics and automation (ICRA)*, Leuven, Belgium, 16–20 May 1998, pp. 3431–3436. IEEE.

Cox IJ (1993) A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision (IJCV)* 10(1): 53–66.

Cristianini N and Shawe-Taylor J (2000) *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge: Cambridge University Press.

Csat L and Opper M (2002) Sparse on-line Gaussian processes. *Neural Computation* 14(3): 641–668.

Dalal N and Triggs B (2005) Histograms of oriented gradients for human detection. In: *Computer vision and pattern recognition*, San Diego, CA, USA, 25 June 2005, pp. 886–893. IEEE.

Dekel O (2008) From online to batch learning with cutoff-averaging. In: *Advances in neural information processing systems (NIPS)*.

Dima C (2006) *Active learning for outdoor perception*. Technical report. Report no. 06–28, Carnegie Mellon University, Pittsburgh, PA.

Dongara JJ, Moler CB, Bunch JR, et al. (1979) *LINPACK's User Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Ellis T and Xu M (2001) Object detection and tracking in an open and dynamic world. In: *Workshop on performance evaluation of tracking and surveillance*.

Ess A, Leibe B and van Gool L (2007) Depth and appearance for mobile scene analysis. In: *International conference on computer vision*.

Ess A, Leibe B, Schindler K, et al. (2009) Moving obstacle detection in highly dynamic scenes. In: *International conference on robotics and automation (ICRA)*.

Finlayson GD, Hordley SD, Lu C, et al. (2002) Removing shadows from images. In: *European conference on computer vision (ECCV)*.

Fischler MA and Bolles RC (1981) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6): 381–395.

Friedman N and Russell S (1997) Image segmentation in video sequences: A probabilistic approach. In: *13th conference on uncertainty in artificial intelligence*.

Ghahramani Z (2004) Unsupervised learning. In: *Advanced Lectures on Machine Learning*, Volume 3176. Berlin/Heidelberg: Springer, pp. 72–112.

Gibbons JD and Chakraborti S (2003) *Nonparametric Statistical Inference* (*Statistics: A Series of Textbooks and Monographs*). Boca Raton, FL: CRC.

Guizilini V (2013) *Non-parametric learning for monocular visual odometry*. PhD Thesis, The University of Sydney, Australia.

Guizilini V and Ramos F (2012) Semi-parametric models for visual odometry. In: *Proceedings of the international conference on robotics and automation (ICRA)*.

Guizilini V and Ramos F (2013) Semi-parametric learning for visual odometry. *The International Journal of Robotics Research* 32(5): 526–546.

Guo G, Li SZ and Chan K (2000) Face recognition by support vector machines. In: *Proceedings of the IEEE international conference on automatic face and gesture recognition*.

Gupta MR and Chen Y (2011) Theory and use of the EM algorithm. *Foundations and Trends in Signal Processing* 4(3): 223–296.

Hammond DK and Simoncelli EP (2007) A machine learning framework for adaptive combination of signal denoising methods. In: *International conference on image processing (ICIP)*.

Han KM and DeSouza GN (2011) Geolocation of multiple targets from airborne video without terrain data. *Journal of Intelligent and Robotics Systems* 62(1): 159–183.

Hartigan JA and Wong MA (1979) A *k*-means clustering algorithm. *JSTOR: Applied Statistics* 28(1): 100–108.

Hartley RI and Zisserman A (2004) *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press.

Hinton G and Sejnowski TJ (1999) *Unsupervised Learning: Foundations of Neural Computation*. Cambridge, MA: The MIT Press.

Horn BKP and Schunck BG (1980) Determining optical flow. *AI Memo No. 572*.

Ibrahim AWN, Ching PW, Seet GLG, et al. (2010) Moving objects detection and tracking framework for UAV-based surveillance. In: *Pacific-rim symposium on image and video technology (PSIVT)*.

Kanungo T, Mount D, Netanyahu N, et al. (2002) An efficient *k*-means clustering algorithm: Analysis and implementation. In: *Pattern analysis and machine intelligence (PAMI)*.

Keck M Jr, Davis J and Tyagi A (2006) Tracking mean shift clustered point clouds for 3D surveillance. In: *Proceedings of the 4th ACM international workshop on video surveillance and sensor networks*.

Kivinen J, Smola AJ and Williamson RC (2003) Online learning with kernels. *IEEE Transactions on Signal Processing* 52(8): 2156–2176.

Koller D, Weber J, Huang T, et al. (1994) Towards robust automatic traffic scene analysis in real-time. In: *Proceedings of the international conference on pattern recognition*.

Leibe B, Leonardis A and Schiele B (2008) Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision* 77(3): 259–289.

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2): 91–110.

Maron O and Ratan AL (1998) Multiple-instance learning for natural scene classification. In: *International conference on machine learning (ICML)*.

MATLAB (2011) Version 7.13 (R2011b). The MathWorks Inc, Natick, MA.

Mohri M, Rostamizadeh A and Talwalkar A (2012) *Foundations of Machine Learning*. Cambridge, MA: The MIT Press.

Monnet A, Mittal A, Paragios N, et al. (2003) Background modelling and subtraction of dynamic scenes. In: *International conference on computer vision*.

Namdev RK, Kundu A, Krishna KM, et al. (2012) Motion segmentation of multiple objects from a freely moving monocular camera. In: *International conference on robotics and automation (ICRA)*.

Neal RM (1996) *Bayesian Learning for Neural Networks*. New York, NY: Springer-Verlag.

Nedevschi S, Danescu R, Frentiu D, et al. (2004) High accuracy stereovision approach for obstacle detection on non-planar roads. *IEEE Proceedings on Intelligent Engineering Systems* 14: 211–216.

Nister D, Naroditsky O and Bergen J (2006) Visual odometry for ground vehicle applications. *Journal of Field Robotics* 23(1): 3–20.

Oliver N, Rosario B and Pentland A (2000) A Bayesian computer vision system for modeling human interactions. *Transactions on Pattern Analysis and Machine Intelligence* 22(8): 831–843.

Osborne MA, Roberts SJ, Rogers A, et al. (2008) Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In: *Proceedings of the 7th international conference on information processing in sensor networks*.

Platt JC (2000) Probabilities for SV machines. In: *Advances in Large Margin Classifiers*. MIT Press, pp. 61–74.

Ranganathan A and Yang MH (2008) Online sparse matrix Gaussian process regression and vision applications. In: *Proceedings of the 10th European conference on computer vision: Part I*, pp. 468–482.

Rasmussen CE and Williams KI (2006) *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press.

Rosenberg A and Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In: *Joint conference on empirical methods in natural language processing and computational natural language learning*.

Saxena A, Sun M and Ng AY (2009) Make3D: Learning 3D scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(5): 824–840.

Scholkopf B and Smola AJ (2002) *Learning with Kernels*. Cambridge, MA: The MIT Press.

Settles B (2010) *Active learning literature survey.* Technical report. Report no. 1648, University of Wisconsin at Madison, WI.

Sheikh Y and Shah M (2005) Bayesian modelling of dynamic scenes for object detection. *Transactions on Pattern Analysis and Machine Intelligence* 27(11): 1778–1792.

Smith M, Posner I and Newman P (2010) Efficient non-parametric surface representations using active sampling for push broom laser data. In: *Proceedings of robotics: Science and systems*.

Sofman B, Lin E, Bagnell JA, et al. (2006) Improving robot navigation through self-supervised online learning. *Journal of Field Robotics* 23(11–12): 1059–1075.

Soga M, Kato T, Ohta M, et al. (2005) Pedestrian detection with stereo vision. In: *IEEE proceedings of the international conference on data engineering*.

Spinello L, Triebel R and Siegwart R (2008) Multimodal people detection and tracking in crowded scenes. In: *Conference on artificial intelligence (physically grounded AI track)*.

Stauffer C and Grimson W (2000a) Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8): 747–757.

Stauffer C and Grimson W (2000b) A probabilistic background model for tracking. In: *European conference on computer vision*.

Stenger B, Ramesh V, Paragios N, et al. (2000) Topology free hidden Markov models: Application to background modeling. In: *European conference on computer vision*.

Szoke I, Lazea G, Tamas L, et al. (2009) Path planning and dynamic objects detection. In: *International conference on advanced robotics (ICAR)*.

Taluker A and Matthies L (2004) Real-time detection of moving objects from moving vehicles using dense stereo and optical flow. In: *Intelligent robots and systems (IROS)*.

Teutsch M and Kruger W (2012) Detection, segmentation and tracking of moving objects in UAV videos. In: *International conference on advanced video and signal-based surveillance (AVSS)*.

Tomasi S and Tomasi C (1994) Good features to track. In: *IEEE computer vision and pattern recognition conference (CVPR)*.

Toyama K, Krumm J, Brumitt B, et al. (1999) Wallflower: Principles and practice of background maintenance. In: *International conference on computer vision*.

Van der Merwe R, de Freitas N, Doucet A, et al. (2001) The unscented particle filter. In: *Advances in neural information processing systems*.

Vapnik VN (1995) *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag.

Wahba G (1990) *Spline Models for Observational Data*. Philadelphia: Society for Industrial and Applied Mathematics (SIAM).

Wang L, Zhao G, Cheng L, et al. (2011) *Machine Learning for Vision-Based Motion Analysis: Theory and Techniques*. London: Springer-Verlag.

Welch G and Bishop G (1995) An introduction to the Kalman filter. Report, University of North Carolina at Chapel Hill, USA.

Williams CKI (1998) Computation with infinite neural networks. *Neural Computation* 10: 1203–1216.

Wren C, Azarbayejani A, Darrel T, et al. (1997) Pfinder: Real time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(7): 780–785.

Yang MH and Belongie S (2009) A family of online boosting algorithms. In: *International conference on computer vision (ICCV)*.

Zhao L and Thorpe C (2000) Stereo and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems* 1(3): 148–154.

Zhong J and Sclaroff S (2003) Segmenting foreground objects from a dynamic textured background via a robust Kalman filter. In: *International conference on computer vision*.

Zhu X (2006) Semi-supervised learning literature survey. Report, University of Wisconsin, USA.

Appendix: Index to Multimedia Extension

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org, after 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table of Multimedia Extension

Extension	Type	Description
1	Video	Real-time results obtained using the proposed algorithm in an urban environment