Online Self-Supervised Segmentation of Dynamic Objects

Vitor Guizilini and Fabio Ramos
Australian Centre for Field Robotics, School of Information Technologies
University of Sydney, Australia
{v.guizilini;f.ramos}@acfr.usyd.edu.au

***Abstract*— We address the problem of automatically segmenting dynamic objects in an urban environment from a moving camera without manual labelling, in an online, self-supervised learning manner. We use input images obtained from a single uncalibrated camera placed on top of a moving vehicle, extracting and matching pairs of sparse features that represent the optical flow information between frames. This optical flow information is initially divided into two classes, static or dynamic, where the static class represents features that comply to the constraints provided by the camera motion and the dynamic class represents the ones that do not. This initial classification is used to incrementally train a Gaussian Process (GP) classifier to segment dynamic objects in new images. The hyperparameters of the GP covariance function are optimized online during navigation, and the available self-supervised dataset is updated as new relevant data is added and redundant data is removed, resulting in a near-constant computing time even after long periods of navigation. The output is a vector containing the probability that each pixel in the image belongs to either the static or dynamic class (ranging from 0 to 1), along with the corresponding uncertainty estimate of the classification. Experiments conducted in an urban environment, with cars and pedestrians as dynamic objects and no prior knowledge or additional sensors, show promising results even when the vehicle is moving at considerable speeds (up to 50 km/h). This scenario produces a large quantity of featureless regions and false matches that is very challenging for conventional approaches. Results obtained using a portable camera device also testify to our algorithm's ability to generalize over different environments and configurations without any fine-tuning of parameters.**

I. Introduction

A truly autonomous robot requires a precise knowledge of the environment around it in order to perform tasks such as path planning, obstacle avoidance and goal-oriented navigation. 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. 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.

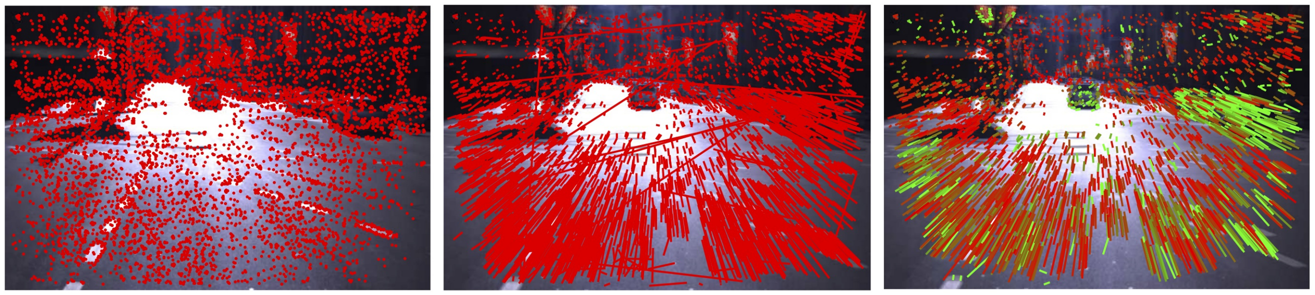
Of all sensors, cameras are becoming increasingly popular because they are relatively inexpensive, small, information-rich, of easy installation and have a wide field of view both horizontally and vertically. The texture and color information provided by visual sensors is also invaluable in situations where a better characterization of objects is necessary, such as road segmentation, lane detection and multi-tracking.

Non-parametric Bayesian techniques have been largely utilized in computer vision applications [27], [23], and are known to provide robust models based on scarce and noisy datasets. This characteristic makes them specially attractive in a real-world scenario, 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, these techniques are capable of generalizing over small variations in the object's appearance and extract the properties that truly represent it, in a way that no specific model could hope to achieve.

The main objective of this paper is to segment dynamic objects from a static background using only a single uncalibrated camera placed on top of a mobile platform, without any prior knowledge of the environment or extra sensors. We develop a novel self-supervised algorithm based on Gaussian Processes (GPs) [21], a non-parametric Bayesian technique used to estimate the probability that each pixel in a image belongs to a dynamic object, based on optical flow information obtained from sparse features. An initial classification is performed using the 7-Point RANSAC algorithm [8], and the results are used as input for the GP to perform a classification on the entire image, along with the corresponding uncertainty estimates. The GP hyperparameters are optimized online during navigation, as new data becomes available, and the current model is constantly updated to incorporate relevant information and remove redundant information as to maintain the computational cost roughly constant.

II. Related Work

Several applications of dynamic object segmentation assume a static camera, which implies that any non-dynamic object will maintain its position over time. In this scenario the traditional approach [23] is to statistically model the background, essentially ”filtering it out”, and treat any change in pixel intensity as a potential dynamic object. In other applications, however, the visual sensor is mobile, usually mounted on top of a robotic platform, and therefore it is impossible to separate background and foreground solely by tracking pixel intensity changes, as static objects will expe-



(a) Feature set

(b) Matching set

(c) Initial RANSAC classification (green indicates dynamic objects and red indicates static background

Fig. 1: Stages of the feature parametrization.

rience relative motion due to camera rotation and translation between frames. A straightforward way to segment this sort of image is to model the ground plane and treat everything else as an object [24], 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 [6], where a coupling between object segmentation and scene geometry is maintained using a Bayesian network.

If a significant portion of the environment is assumed static, the relative motion of static objects can be filtered out by calculating the optical flow [11] of the image and using a voting method, such as RANSAC [8], to elect the most probable motion hypothesis. Any region that does not comply to this constraint is assumed to be a dynamic object, and can be tracked using classical approaches such as Extended Kalman Filters or Particle Filters (robust data association algorithms [2] and occlusion-handling techniques [6] are necessary to deal with very cluttered environments).

If more than one camera is available, a stereo triangulation can provide a 3D position estimate for matched features [10], incorporating extra information that could be used to facilitate and improve object clustering and tracking [1]. A more accurate segmentation 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 [12], directly on the camera images [5], [13] or in a combination of both [7]. The static background information can also be used to improve visual odometry applications [18], [9], [6], since its optical flow values now reflect solely the camera's own rotation and translation.

III. Optical Flow Extraction

Our approach uses sparse optical flow information obtained from consecutive pairs of frames during vehicle navigation. 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 SIFT (Scale-Invariant Feature Transformation) [15] and corner detection [25] algorithms, to ensure a high density of features throughout the entire image (Fig. 1a). This feature set is matched with the feature set from the subsequent frame using 128-element SIFT descriptors, and the resulting set is shown in Fig. 1b, where each match is represented as a line segment connecting the corresponding features from each frame. This sparse optical flow information is then classified according to the RANSAC algorithm [8], an iterative non-deterministic method used to estimate the parameters of a mathematical model from a set of observed data which contains outliers. In this context the mathematical model is the Fundamental Matrix [10], defined as a 3×3 matrix F that optimizes the equation

$$\mathbf{u}_2^T F \mathbf{u}_1 = 0, \tag{1}$$

where $\mathbf{u}_i = \{u, v, 1\}$ represents the normalized image coordinates of each feature in frame i. The projection of a feature $\mathbf{u}_1$ in frame 2 is a line with coefficients $F\mathbf{u}_1$ (the epipolar line), and symmetrically the projection of a feature $\mathbf{u}_2$ in frame 1 is a line with coefficients $F^T\mathbf{u}_2$. Assuming that most of the environment is static, the RANSAC algorithm should elect the Fundamental Matrix that best represents camera motion, and therefore any feature whose match coincides with the epipolar line in that frame should belong to a static object. The further away the corresponding match is to the epipolar line, the higher the chance that it belongs to a dynamic object, and this distance allows for a probabilistic classification of features, ranging from 0 (the corresponding match coincides with the epipolar line) to 1 (the furthest away a corresponding match may be from the epipolar line before it is discarded as false).

An example of this classification is presented in Fig. 1c, where each matched pair is depicted with a color ranging from red (static) to green (dynamic). While it is clear that the RANSAC classification was able to correctly segment most of the dynamic objects in the scene (the car in front of the camera and the ones to its right), there is a substantial amount of static matches that were mistakenly classified as dynamic, specially in the lower portions of the image where the street is represented. These mistakes are mostly due to false matches caused by poor texture, since the street lacks of visual cues that allow for an unambiguous association between descriptors.

This sparse optical flow information will serve as input for the GP framework described in the next section. Even though the SIFT descriptor is highly effective in feature matching between frames, possessing several invariance properties, it does not serve as well in applications that aim for generalization, which is the case in non-parametric modelling based on training data. Because of this, each matched feature receives a new descriptor $\mathbf{x}$ composed of its image coordinates (u, v) and the average of its intensity values (I_M, I_R, I_G, I_B) in a surrounding window for each RGB color and for the corresponding monochromatic M transformation (as a convention, we use the first frame for these calculations). The coordinate parameters are useful in modelling how different regions of the image react to changes in optical flow values, and the intensity parameters are useful in modelling the different objects in the same region of the image and their boundaries (other relevant parameters may be added seamlessly without any further changes in the algorithm).

The final sparse optical flow information that represents each frame is then of the form $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is a $D \times N$ matrix containing the $D = 6$ descriptor values $\mathbf{x}_n = \{u, v, I_M, I_R, I_G, I_B\}_n$ 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 belong to a dynamic object, according to the RANSAC algorithm.

IV. Gaussian Process Classification

From a machine learning perspective, the problem of dynamic object segmentation can be seen as a self-supervised classification problem, where each pixel in the image has a certain probability of either belonging to a static or dynamic object. We use a Gaussian Process [21] to perform this classification based on initial 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 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.

A. Inference Equations and Covariance Function

A Gaussian Process 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 $\Lambda = (\mathbf{X}, \mathbf{y}) = \{\mathbf{x}, y\}_{n=1}^{N}$ 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 a 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 a positive-definite kernel (the covariance function) 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:

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

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

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 behaviour of a typical segmented image (sudden changes from a static to a dynamic object), we choose here the neural network covariance function [28] to model the relationship between input data. 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 [17] as:

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

where $\widetilde{\mathbf{x}} = (1, x_1, \ldots, x_D)^T$ 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 (here assumed diagonal with $D + 1$ eigenvalues). The expression also contains a sigmoid-like function, $arcsin(x)$, to model sharp transitions and non-linearities.

B. Hyperparameter Optimization

During the training stage, the hyperparameters of the covariance function are optimized as to minimize a certain cost function, here chosen to be the log-marginal likelihood [21] due to its ability to balance data fit and model complexity, thus minimizing the chance of over-fitting. 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 approach 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.

C. 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

Gaussian Process is its ability to also calculate the variance $\mathcal{V}(\overline{f}_*)$ of each prediction, that 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 [21], 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 [20], and the implemented version for training its parameters is defined as:

$$p(y_i|\mathbf{X}, y_{-i}, \theta) = \Phi\left(\frac{y_i(\alpha\mu_i + \beta)}{1 + \alpha^2\sigma_i^2}\right), \tag{6}$$

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 of recalculating the entire covariance matrix for each training point [26]. The new expressions for the LLO predictive mean and variance are presented in Eq. 7, and they allow the classification of each pixel in the image as a static, dynamic or unsure object, according to user-defined thresholds:

$$\mu_i = y_i - \frac{[K^{-1}y]_i}{[K^{-1}]_{ii}} \qquad \sigma_i^2 = \frac{1}{[K^{-1}]_{ii}}. \tag{7}$$

D. Incremental Updates

One drawback of Gaussian Processes is the cost of inverting K in Eqs. 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 (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 [19], [22] is using the Cholesky decomposition to calculate the predictive mean $\overline{f}_*$ and variance $\mathcal{V}(\overline{f}_*)$ in Eqs. 3 and 4, instead of 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. It works by obtaining a lower triangular matrix L, called the Cholesky factor, such that $L^T L = K$. The marginalization (removal) of the second row and column of a hypothetical 3×3 covariance matrix is calculated as:

$$K = \begin{bmatrix} K_{1,1} & \mathbf{k}_{1,2} & K_{1,3} \\ \mathbf{k}_{1,2}^T & k_{2,2} & \mathbf{k}_{2,3} \\ K_{1,3}^T & \mathbf{k}_{2,3}^T & K_{3,3} \end{bmatrix} \quad L = \begin{bmatrix} C_{1,1} & \mathbf{c}_{1,2} & C_{1,3} \\ \mathbf{0} & c_{2,2} & \mathbf{c}_{2,3} \\ \mathbf{0} & \mathbf{0} & C_{3,3} \end{bmatrix} \tag{8}$$

$$L' = \begin{bmatrix} C_{1,1} & C_{1,3} \\ \mathbf{0} & \gamma(C_{3,3}^T C_{3,3} + \mathbf{c}_{2,3}^T \mathbf{c}_{2,3}) \end{bmatrix},$$

where γ is the Cholesky update defined in [16], which exploits the special structure of $\mathbf{c}_{2,3}^T\mathbf{c}_{2,3}$ and is of complexity $\mathcal{O}(n^2)$. Similarly, the incorporation of a new point $\mathbf{x}_*$ into a 2×2 covariance matrix is calculated as shown in Eq. 9, where $K_{3,3} = k(\mathbf{x}_*, \mathbf{x}_*)$ and $\mathbf{c}_3$ is the solution of the linear system $L\mathbf{c}_3 = \mathbf{k}_{1,2}$:

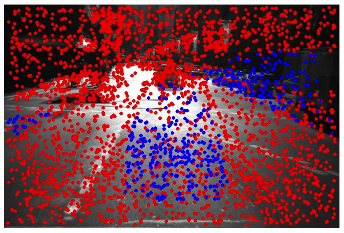
$$K = \begin{bmatrix} K_{1,1} & \mathbf{k}_{1,2} \\ \mathbf{k}_{1,2}^T & K_{2,2} \end{bmatrix} \quad L = \begin{bmatrix} C_{1,1} & C_{1,2} \\ \mathbf{0} & C_{2,2} \end{bmatrix}$$

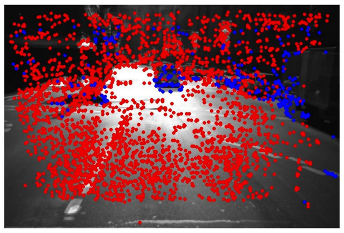
$$L' = \begin{bmatrix} \begin{matrix} C_{1,1} & C_{1,2} \\ & C_{2,2} \end{matrix} & \mathbf{c}_3 \\ \mathbf{0} & chol(K_{3,3} - \mathbf{c}_3^T\mathbf{c}_3) \end{bmatrix}. \tag{9}$$

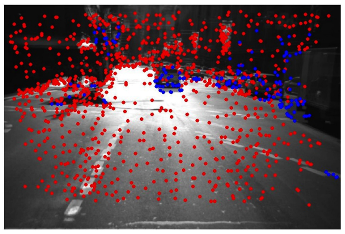
E. Feature Filtering

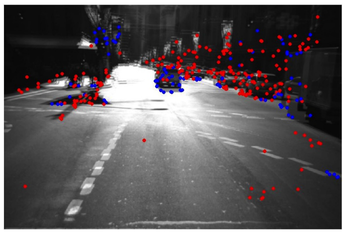
The feature extraction and matching techniques described in the previous section produce an average of around 7000 features per frame, and it is impractical to incorporate all these new points to the GP framework at every iteration, as it would create an exceedingly complex and computationally prohibitive model after 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 could be safely discarded without compromising model accuracy. Also, it is necessary to discard erroneous information generated by RANSAC misclassifications, that could skew results. In order to determine which data points should be incorporated and removed from the GP framework at each iteration, we impose a series of filtering steps over different subsets of the available data, designed to remove redundant and unreliable information from the non-parametric model:

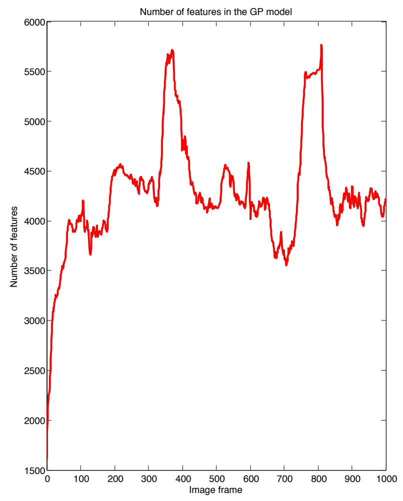
1) A density constraint is imposed on the input information, discarding points whose classification is similar to their closest neighbours according to a certain distance threshold. This step decreases the amount of data available for potential incorporation to the non-parametric model, while still maintaining its spacial distribution on the input space.
2) An inference is performed on the remaining new data points, classifying them according to the current non-parametric model (all new data points are added at the first iteration of the algorithm). All those incorrectly classified are added to the covariance matrix K and its corresponding Cholesky decomposition L, increasing the amount of information in the non-parametric model. Those correctly classified are discarded, since


(a) Current model


(b) New feature set


(c) Filtered feature set


(d) Added feature set



(e) Number of features used by the non-parametric model

Fig. 2: Online non-parametric model update. On figures (a)-(d) red dots indicate a static classification and blue dots indicate a dynamic classification, according to the RANSAC algorithm.

their position on the input space is already well-defined.

3) An inference is performed now on the non-parametric model itself, removing data points that were incorrectly classified even after their incorporation. This step is important in eliminating RANSAC misclassifications, as they are assumed to be a minority and therefore less representative of their position on the input space.
4) Finally, we apply the density constraint also on the non-parametric model, removing data points with similar classification as their closest neighbours. As in Item 1, this step decreases the amount of data available for the non-parametric model while still maintaining its spatial distribution on the input space.

In combination, these four steps comprise around 35% of the total computational time of the algorithm, mostly due to the inference steps necessary to determine which data points are incorrectly classified and which ones share a similar classification that of its neighbours. However, empirical tests show that they reduce the amount of information processed by the algorithm by a factor of 10, by discarding redundant data points that would dramatically increase computational time (a GP is of complexity $\mathcal{O}(n^3)$, and the online update is of complexity $\mathcal{O}(n^2)$).

Since we are dealing with a finite-size input space (the D-dimensional descriptive vector representing each matched feature is limited by image size and intensity values saturation), as navigation continues the number of data points forming the non-parametric model eventually stabilizes at around 4000 (see Fig. 2e). Sudden increases in data point incorporation indicate changes in environment structure, where the algorithm has to learn the behaviour of new objects and adapt to changes in old ones (i.e. a stationary car starts moving), however these spikes soon cease and the non-parametric model returns to its original stable size.

V. Experimental Results

The proposed algorithm was tested in both an urban scenario, using data collected from a modified vehicle equipped with a single uncalibrated camera, and from a pedestrian perspective, using data collected from a portable camera device. The urban scenario results were compared with other approaches to automatic dynamic object segmentation, as a way to validate how our algorithm improves over already existing techniques, and the portable camera device results are used to show how our algorithm is capable of generalizing over different environments and configurations without fine-tuning its parameters.

A. *Urban Scenario*

The urban dataset is composed of 14500 images obtained at a rate of 3 frames per second in 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 global changes in luminosity, due to cloud coverage and tall buildings, and also to a wide variation of structures around the vehicle, as it was driven around 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 is incorporated the computational time stabilized at a few seconds

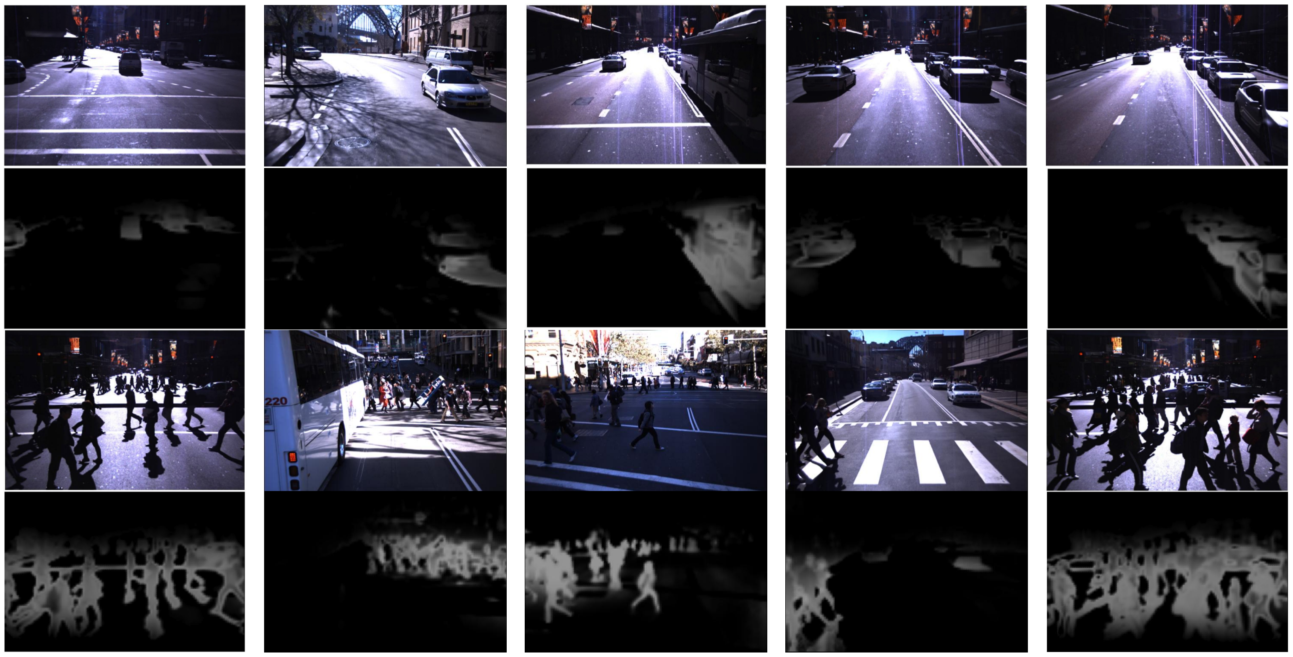


Fig. 3: Automatic dynamic object segmentation results. Each pixel has a color ranging from black (static) to white (dynamic), according to its probability of belonging to either class.

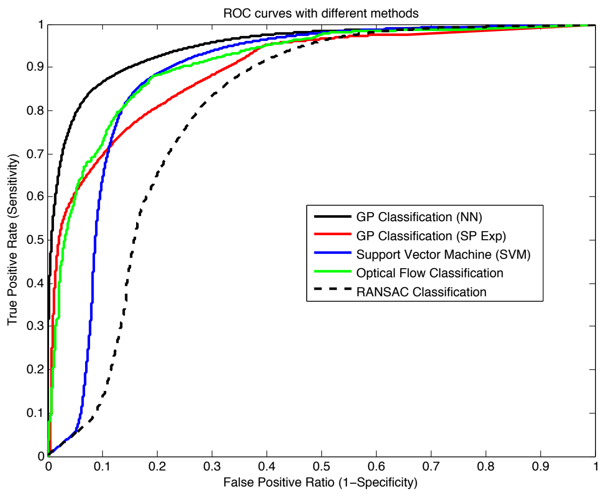
per frame, mostly due to the frequent inferences necessary to decide which data points should be added/removed from the non-parametric model. Approximation techniques, such as Sparse Gaussian Processes [4] or Subset of Regressors [21] would substantially improve computational speed, since it is natural to assume that distant features should have no impact in calculations.

The classification results obtained in different instants of navigation are presented in Fig. 3,where lighter areas represent dynamic objects. The inference was conducted at every 5th pixel of the image, both horizontally and vertically, and the intermediary pixels were calculated accordingly using linear interpolation. These results testify to the algorithm's ability to correctly segment dynamic objects from a static background with high precision, even when dealing with a crowd or in situations where the camera is moving at relatively high speeds. The GP framework was able to remove virtually all the false matches related to the street on the lower portions of the image, which was a major concern during the RANSAC classification (see Fig. 1c). We attribute this success to the descriptor used to represent each feature in the input space, that contains color intensity information in addition to coordinate values, and this information allowed the non-parametric model to correctly correlate street features and reclassify the outliers as part of a static object.

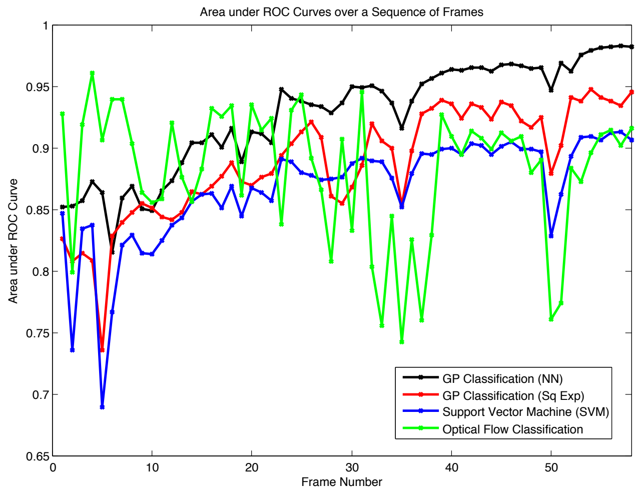
Finally, the proposed algorithm was compared with three other approaches to dynamic object segmentation and the results are presented in Fig 4, (based on information from 100 hand-labelled images collected in different instants of navigation). The dotted line represents the initial classification results obtained using RANSAC, the black line indicates the proposed framework, and the red line indicates the same framework but with the neural network covariance function substituted by the square-exponential. The blue line represents the use of Support Vector Machine (SVM) [3] as the self-supervised classification method instead of GPs, and the green line are the Optical Flow Classification (OFC) results obtained based on [14].

The ROC curves for each approach are shown in Fig. 4a, where it is possible to see that the proposed method outperforms the others in all threshold levels, and in particular that it improves on the initial RANSAC classification results by a significant margin, without incorporating any extra information. It is also possible to see the importance of covariance function selection, since the same framework performed significantly worse when the square-exponential covariance function was used. Fig 4b depicts the area under the ROC curve for each frame at the beginning of navigation, indicating how accuracy changes as new data is incorporated to the model.

As expected, the OFC does not improve over time, since it is not based on learning techniques, using instead individual information from each frame. The accuracy of the three other approaches increase steadily over time, with occasional drops that indicate moments where there is a significant change in the environment (the camera started moving, or a new object entered its field of vision), and the proposed approach constantly outperforms the other two. It is interesting to see that at the beginning of navigation the OFC is the best solution, since there was no time for the non-parametric model to learn the environment characteristics, but after a few frames the proposed method improves and becomes the best solution, while the OFC oscillates heavily at each iteration.



(a) ROC curves



(b) Classification accuracy during navigation

Fig. 4: Comparison between different methods of dynamic object segmentation. The SVM Classification was obtained using the same framework presented here but with Support Vector Machine [3] instead of Gaussian Processes, and the Optical Flow Classification was based on [14].

B. *Portable camera device*

The same algorithm, without any further modifications, was also tested in images obtained using a portable camera device (Fig. 5), 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 is much smaller and its motion is unconstrained by two-dimensional vehicle dynamics. The results obtained using this configuration are presented in Fig. 6, where it is possible to see that again there is a wide variation in luminosity and types of structures, and the algorithm was still capable of correctly segmenting most of the dynamic objects in each scene.

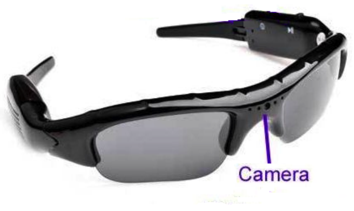


Fig. 5: Portable camera device.

Still, it is possible to see some deviations on the segmentation results, mostly caused by the presence of shadows, which are classified as dynamic objects because of the purely visual nature of the segmentation technique, and the non-detection of far-away dynamic objects, due to their low relative speeds that are dismissed as noise. Sudden changes in texture and color also generate gaps in the segmentation results, because the high-dimensional input vector used by the GP uses both 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-labelled image information collected from a single uncalibrated camera.

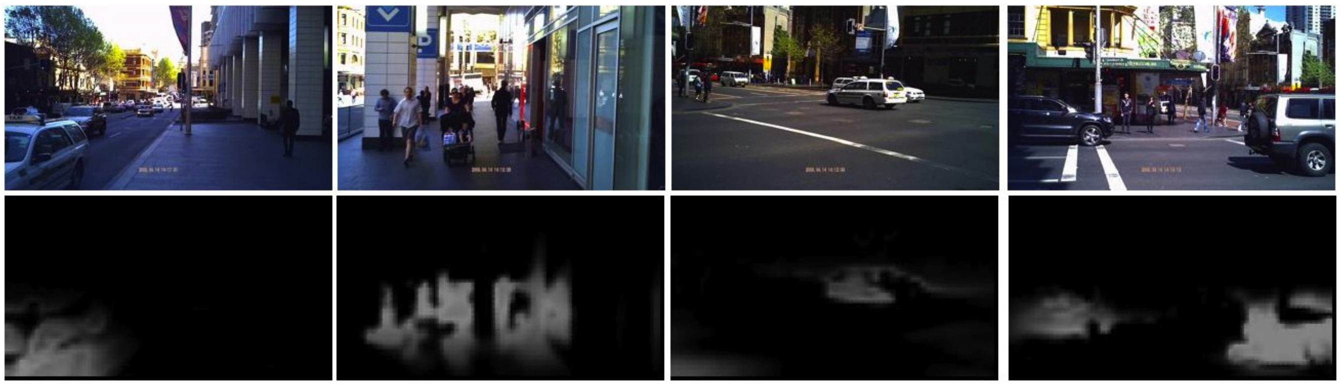


Fig. 6: Experiments with the portable camera device.

VI. Conclusion

This paper presented a novel approach to automatic dynamic object segmentation with a single mobile uncalibrated camera, based on self-supervised learning. Sparse matched features were sorted according to the RANSAC algorithm to generate an initial classification between a static background and dynamic foreground objects, and this initial classification was refined in an online self-supervised manner using a Gaussian Process, a non-parametric Bayesian inference technique. Experiments were conducted in an urban scenario, with data collected from a modified vehicle navigating at speeds of up to 50 km/h and interacting with pedestrians, cars and buses, and show promising results, with the proposed algorithm outperforming other dynamic object segmentation approaches.

Since it relies purely on visual information, the 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 size and low relative speeds, which difficults the process of 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 learn its visual information in different scales and viewpoints. Furthermore, the learning aspect of this framework could also in principle allow the self-supervised creation of a library of segmented objects, extending the initial binary classification into a more semantically relevant territory. Additionally, the introduction of a library of segmented objects creates the possibility of a "potentially dynamic" class, where a certain object is inherently dynamic but is currently stationary. Future work will focus on algorithm speed, exploring assumptions such as sparsity to decrease computational cost and allow real-time model update and inference.

References

[1] Almanza-Ojeda, D.-L., and Ibarra-Manzano, M.-A. 3d visual information for dynamic objects detection and tracking during mobile robot navigation. *Recent Advances in Mobile Robotics* (2011).

[2] Cox, I. J. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision (IJCV)* (1993).

[3] Cristianini, N., and Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[4] Csat, L., and Opper, M. Sparse on-line gaussian processes. Tech. rep., Massachusetts Institute of Technology, 2002.

[5] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. *Conference on Computer Vision and Pattern Recognition* (2005).

[6] Ess, A., Leibe, B., Schindler, K., and Gool, L. V. Moving obstacle detection in highly dynamic scenes. *International Conference on Robotics and Automation (ICRA)* (2009).

[7] Ess, A., Leibe, B., and van Gool, L. Depth and appearance for mobile scene analysis. *International Conference on Computer Vision* (2007).

[8] Fischler, M. A., and Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* (1981).

[9] Guizilini, V., and Ramos, F. Semi-parametric models for visual odometry. In *Proc. Int. Conference on Robotics and Automation (ICRA)* (2012).

[10] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[11] Horn, B. K. P., and Schunck, B. G. Determining optical flow. *AI Memo No. 572* (1980).

[12] Jr., M. K., Davis, J., and Tyagi, A. Tracking mean shift clustered point clouds for 3d surveillance. *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks* (2006).

[13] Leibe, B., Leonardis, A., and Schiele, B. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision* (2008).

[14] Liu, C. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[15] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* (2004).

[16] MATLAB. *version 7.13 (R2011b)*. The MathWorks Inc., 2011.

[17] Neal, R. M. *Bayesian Learning for Neural Networks*. Springer-Verlag New York Inc., 1996.

[18] Nister, D., Naroditsky, O., and Bergen, J. Visual odometry for ground vehicle applications. *Journal of Field Robotics* (January 2006).

[19] Osborne, M. A., Roberts, S. J., Rogers, A., Ramchurn, S. D., and Jennings, N. R. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. *Proceedings of the 7th International Conference on Information Processing in Sensor Networks* (2008).

[20] Platt, J. C. *Probabilities for SV Machines*. Advances in Large Margin Classifiers, 2000.

[21] Rasmussen, C. E., and Williams, K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[22] Scholkopf, B., and Smola, A. J. *Learning with Kernels*. The MIT Press, 2002.

[23] Sheikh, Y., and Shah, M. Bayesian modelling of dynamic scenes for object detection. *Transactions on Pattern Analysis and Machine Intelligence* (2005).

[24] Soga, M., Kato, T., Ohta, M., and Ninomiya, Y. Pedestrian detection with stereo vision. *IEEE Proceedings on the International Conference on Data Engineering* (2005).

[25] Tomasi, S., and Tomasi, C. Good features to track. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR)* (1994).

[26] Wahba, G. Spline models for observational data. *Conference Series in Applied Mathematics* (1990).

[27] Wang, L., Zhao, G., Cheng, L., and PietikŁinen, M. *Machine Learning for Vision-Based Motion Analysis: Theory and Techniques*. Advances in Pattern Recognition, 2011.

[28] Williams, C. K. I. Computation with infinite neural networks. *Neural Computation* (1998).