Video Segmentation Based on Motion Coherence of Particles in a Video Sequence

Luciano S. Silva and Jacob Scharcanski, *Senior Member, IEEE*

***Abstract*—This work describes an approach for object-oriented video segmentation based on motion coherence. Using a tracking process based on adaptively sampled points (namely, particles), 2-D motion patterns are identified with an ensemble clustering approach. Particles are clustered to obtain a pixel-wise segmentation in space and time domains. The segmentation result is mapped to an image spatio-temporal feature space. Thus, the different constituent parts of the scene that move coherently along the video sequence are mapped to volumes in this spatio-temporal space. These volumes make the redundancy in the temporal sense more explicit, leading to potential gains in video coding applications. The proposed solution is robust and more generic than similar approaches for 2-D video segmentation found in the literature. In order to illustrate the potential advantages of using the proposed motion segmentation approach in video coding applications, the PSNR of the temporal predictions and the entropies of prediction errors obtained in our experiments are presented, and compared with other methods. Our experiments with real and synthetic sequences suggest that our method also could be used in other image processing and computer vision tasks, besides video coding, such as video information retrieval and video understanding.**

***Index Terms*—Ensemble clustering, motion segmentation, object-based video segmentation, point tracking, video coding.**

I. Introduction

MOTION segmentation is an important preprocessing step in many computer vision and video processing tasks, such as surveillance, object tracking, video coding, information retrieval, and video analysis. These applications motivated the development of several 2-D motion segmentation techniques, where each frame of a video sequence is split into regions that move coherently. By 2-D motion, here we denote the motion of objects in a 3-D scene projected in the image plane of a camera. However, 2-D motion segmentation often leads to video over-segmentation. For example, a scene composed by static rigid objects, captured by a moving camera, can be over-segmented in several 2-D motion regions due to several reasons, such as depth discontinuities, occlusions, and the perspective projection effect. In some situations, a scene comprises several moving objects and it is necessary to identify each object as a coherent motion entity. In these cases, the segmentation process can be approached by considering the objects as moving in 3-D space (x, y, z), and this approach has motivated several works in 3-D motion segmentation [1]–[6]. Spatio-temporal motion segmentation [7]–[10] is a different approach, where different moving objects are segmented in volumes (called tunnels [10]) in the domain formed by the spatial dimensions (e.g., x, y) and the temporal dimension (t), and these volumes are delimited by object motion boundaries (i.e., motion discontinuities). The definition of object in a video segmentation framework is related to the concept of region homogeneity, and different applications require different region homogeneity criteria. In video coding, for example, segmentation is frequently used to explore the data redundancy in time [11]. In this context, an object region that retains its characteristics (e.g., color or texture) along the sequence can be considered homogeneous and redundant. Thus, even if the object region moves along the temporal sequence, the region representation remains the same, i.e., redundant, within the object motion boundaries.

In 3-D motion segmentation, the concept of object is related to actual objects existing in 3-D space that do not change their 3-D characteristics over time. The concept of object in spatio-temporal segmentation is different, since an object is represented by a spatio-temporal tunnel formed by a sequence of 2-D projections, each 2-D projection obtained at a time t of an object in 3-D space. Two sets of parameters are often used to describe 3-D parametric motion, the set of global parameters representing the camera and/or object motion, and the set of local parameters representing the object attributes (e.g., shape, color, and texture). However, the estimation of a large number of parameters often is awkward, particularly in the presence of noise and outliers. An outlier can be, for example, a point trajectory incorrectly computed. On the other hand, when camera translations and depth variations are small compared to the distance of the camera to the scene objects, simpler 2-D motion models can become attractive [12]. In 2-D parametric motion segmentation, a small number of parameters is needed to describe the object motion, making the motion segmentation more robust to noise. Although the computer vision community has been consistently working towards improving 3-D motion segmentation [13]–[20], 2-D motion segmentation also has received attention, since it still has some open issues and it is suitable for some important video processing tasks like video coding, where a simple representation is important, and in general the semantic aspects of the scene are less relevant [11].

In the context of motion estimation, the literature can be divided in two classes of methods: direct methods [12] and feature-based methods [21]. Motion segmentation methods

Manuscript received November 12, 2008; revised August 17, 2009. First published December 18, 2009; current version published March 17, 2010. This work was supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil (CNPq). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Amy R. Reibman.

The authors are with the Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil 91501-970 (e-mail: issilva@inf.ufrgs.br; jacobs@inf.ufrgs.br).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2009.2038778

can also be divided according to these two paradigms. Direct methods recover the unknown parameters directly from measurable image quantities at each pixel in the image, solving two problems simultaneously: 1) the motion of the camera and/or objects of the scene, and 2) the correspondence of every pixel. This is in contrast with the feature-based methods, which first extract a sparse set of distinct features from each image separately, and then recover and analyze their correspondences in order to determine motion. Feature-based methods minimize an error measure that is based on distances between a few corresponding features, while direct methods minimize a global error measure that is based on direct image information collected from all pixels in the image. For this reason, direct methods are sometimes called dense methods in the literature. However, we should note that, according to the feature-based philosophy, motion can be estimated using sparse features in a first step, and in a second step the motion should guide the dense correspondence for the nonfeature pixels. Thus, in this work we refer to any motion estimation/segmentation method that yields correspondence/classification for each pixel as "dense", even if the motion estimation/segmentation core is guided only by a sparse set of features. It is important to observe that with direct methods the pixel correspondence/classification is performed directly with the measurable image quantities at each pixel, while in feature-based methods this is done indirectly, based on independent feature measurements in a set of sparse pixels. An important property of the direct methods is that they can sucessfully estimate global motion even in the presence of multiple motions and/or outliers [12]. However, computational time is wasted by including in the minimization a large number of pixels where no flow can be reliably estimated. Moreover, normal flows can only be combined across regions of the image that have some simple parametric form (such as an affine or quadratic [21]), and motion estimate errors can accumulate when frames are distant apart and the data may not fit the model very well. On the other hand, feature-based methods initially ignore areas of low information, resulting in a problem with fewer parameters to be estimated, with good convergence even for long sequences. Further, there is a wide of choice of algorithms to estimate parameters for more complex models (e.g., epipolar or trifocal geometry) from point or line features. Nevertheless, in these methods, feature correspondences are computed independently, being more susceptible to outliers.

Variational frameworks have been widely used in the context of direct motion segmentation [7]–[10], [22]–[24]. In order to improve the quality of object segmentation, shape priors were integrated in variational methods by Cremer *et al.* [22]–[24]. However, prior information about the objects in a sequence often is not available. Therefore, Mitiche *et al.* [7] proposed to segment moving objects by detecting the tunnel delimited by motion discontinuities in the spatio-temporal domain. Feghali and Mitiche further extended this idea to also handle moving cameras [8], and later Sekkati and Mitiche proposed a 3-D direct motion segmentation method with a similar approach in [14] and [25]. They formulated the problem as a Bayesian motion partitioning problem, and approached the corresponding Euler-Lagrange equations as a level-set problem. Cremers and Soatto [9] proposed a multiphase level-set method to segment a video using spatio-temporal surfaces (tunnels) that separate regions with piecewise constant motion. A limitation of segmentation methods based on motion discontinuities (motion boundaries) is that they tend to fail in frames where these boundaries are not evident, or do not exist. For example, a static object in a static background that moves only in the last few frames of the sequence can not be correctly segmented at the beginning of the sequence, since there were no motion boundaries and the object was not moving with respect to the background. Ristivojevic and Konrad [10] also proposed a spatio-temporal segmentation method based on the level-set approach, where they defined the concepts of occlusion volumes (i.e., background regions that become occluded) and exposed volumes (background regions that become visible). The authors suggested that potentially the concepts of occlusion and exposed volumes can be applied in the next-generation of video compression methods. As occlusion and exposed areas are difficult to predict, new efficient compression techniques could be developed by estimating occlusion and exposed areas *a priori.* However, the occlusion volume concept have limitations as proposed, since it does not consider the occlusion of a moving object by the background, or by another moving object. A characteristic shared by most variational methods is that they rely on motion models defined *a priori.* If the data do not fit these models well, the methods tend to fail—except when special conditions can be assumed, like static background, number of objects known *a priori,* etc.

Feature-based methods for motion segmentation usually consist of two independent stages: 1) feature selection and/or correspondence and 2) motion parameter estimation [21]. The second stage often is performed through factorization methods [2], [5], [15], [26], although some simpler clustering strategy can be used [27], [28]. In factorization methods, motion and shape information are treated separately by applying constraints to the scene projection on the image formation plane, as well as on the object shape and motion. Several methods have been proposed for sparse feature selection and/or correspondence, and among the most popular are the Harris Corner Detector [29], KLT [30], and SIFT [31]. As mentioned before, a weak point of these sparse feature-based methods is that feature correspondences are computed independently. Thus, they are very sensitive to outliers, making them susceptible to errors in motion parameter estimation/segmentation. Also, homogeneous regions of a frame may present none or few features, making the motion estimation/segmentation difficult (or even impossible) in large areas of the video frames.

Eventually, we can obtain consistent object segmentation by combining several partial informations about an object. The idea of merging object segmentation information from several parts of a sequence was proposed by Geldon *et al.* [32], as a probabilistic multiple hypothesis tracking (PMHT) approach. The authors propose to track an object over the whole image sequence, by combining partial object segmentations previously computed in different parts of the sequence. This is done by modeling the motion and geometry of the objects, and these models are combined assuming smooth trajectories, and are used to eliminate ambiguities caused by occlusions and incorrect detections. However, the object motion and/or geometry modeling accuracy depends on how well the models fit the data, besides the trajectory constraints preclude the application of this approach in videos with discontinuous object trajectories, which is common in sequences obtained with hand-held cameras, for example.

In this paper, we present a new approach for video object segmentation where objects are defined as nonoverlapping regions (at pixel level) in the spatio-temporal domain. These regions are expected to retain their spatial and photometric characteristics in time. Our approach combines the advantages of dense and feature-based methods, as described next. Initially, correspondences in time of sparse points (i.e., particles) are computed, so that long-range motion patterns can be identified. However, instead of computing point correspondences independently (as done in many feature-based methods), neighboring particles are treated as they were linked, reducing the chance of occurring outliers and avoiding the aperture problem [33].[1] Moreover, the density of sampled points (i.e., particles) is adaptive, and denser particle distributions are used in regions where precision is more important (for example, in motion boundaries), saving computation without neglecting homogeneous regions. To compute particle correspondences in a video sequence, we use the approach proposed by Sand and Teller [34], which relies on particles that are located with sub-pixel precision. After the particle correspondences are computed, particles are clustered in each frame of the sequence. The individual particle clusterings at frame level, are then further grouped in larger sets of particles associated to different frames, according to an ensemble clustering strategy. Finally, a dense video frame representation (i.e., a pixel-wise representation) of the final clustering is obtained. The proposed method is general in the sense that it does not rely on motion models, does not impose trajectory constraints and segments multiple objects of arbitrary shapes, without knowing the number of objects *a priori*. Instead of motion boundaries, the segmentation is guided by the consistent motion behavior of sample points of the frames. This strategy allows to extract longer tunnels in the spatio-temporal domain. Besides, it does not need any special treatment for changes in topology or new objects that arise along the sequence. The proposed method potentially has the ability to generate occluded and exposed volumes, since motion patterns are discovered and associated to each moving region, and voxels of the spatio-temporal volumes are classified as belonging to object volumes, occluded volumes or exposed volumes. The proposed approach generates a simple scene representation, adequate for object video coding, and also delivers a more redundant and temporally persistent partition of the scene than direct video segmentation methods and motion prediction strategies. Experimental results for synthetic and natural video sequences are used to illustrate the properties of the proposed method, and to show its potential in video coding applications.

This paper is organized as follows. Section I discusses the state of the art in video segmentation, and contextualizes our work. An overview of the proposed approach is presented in Section II. In Section III, we describe the estimation of particle trajectories along the video sequence, and the segmentation of particle trajectories is described in Section IV. In Section V, we present the dense representation for the motion segmentation. Section VI presents some experimental results obtained with the proposed approach. Finally, Section VII summarizes the main contributions of this paper, and discusses some limitations of the proposed approach and future work directions.

[1]The aperture problem arises from the ambiguity of 1-D motion viewed through a small aperture. Therefore, locally, we can detect motion only in the orientation perpendicular to the moving object contour, and we can not detect motion in homogeneous regions.

II. Method Overview

The structure of the proposed coherent motion segmentation approach can be divided in three main parts.

1) Estimation of Particle Trajectories (see Section III).
2) Segmentation of Particle Trajectories (see Section IV).
3) Dense Segmentation Extraction (see Section V).

The first part concerns the selection and tracking of a set of points of the scene (namely, particles). This stage takes as input the original video frames, and returns as output a set of particles and their respective trajectories. During the estimation of particle trajectories, the particles whose correspondent point locations in the scene suffer occlusion are eliminated, and new particles are created in regions that become newly visible along the video sequence.

The second part deals with the segmentation of particle trajectories, so that particles moving coherently are grouped together. This stage takes as input the particles trajectories computed in the first stage, and returns labels for all the particles as outputs, representing the motion segmentation of frame regions according to the particle trajectories. The segmentation of particle trajectories can be divided in four steps.

- Clustering of 2-frame motion vectors: in this step, clusterings of particles are performed with displacement motion vectors taken from pairs of frames. Only neighboring frames are considered (1, 2, and 3 unit time distances), and clusterings are computed in an independent way. For each pair of frames, the input to this step is the position of particles in each frame, and the output is a set of particle clusterings and their labels, valid for each pair of frames considered.
- Ensemble clustering of particles: here, all the clusterings computed in the previous step are processed simultaneously to produce a unique division of the full set of particles in sub-sets of particles in coherent motion, called meta-clusters; several sets of clustering labels are taken as input to this step, and a unique set of segmentation labels (several particles in coherent motion share the same segmentation label) are returned as output.
- Meta-clustering validation: in this step, particles that were segmented in the previous step are compared to meta-cluster prototypes in terms of motion and spatial position to detect incorrectly labeled particles and, when this occurs, particles are re-labeled. A set of segmentation labels is taken as input, and a corrected set of segmentation labels is returned as output.
- Spatial filtering: in this step, outliers are eliminated and groups of adjacent particles that are not significant. The particle labels are analyzed spatially, and links between particles are created to define spatial adjacency in each frame. Small groups of adjacent particle labels that are not significant are then re-assigned. This step takes as input a

set of particle labels, and returns as output a filtered set of particle labels.

The third and final part of the proposed motion segmentation method is the dense segmentation extraction. This stage takes as input the original video frames, the segmentation labels returned by the second stage, as well as the particle positions returned by the first stage, and returns as the output the corresponding segmentation labels for each pixel of each frame of the video sequence. This is equivalent to the segmentation of a spatio-temporal volume in several tunnels. The dense segmentation extraction is done by creating implicit functions for each particle, based on motion and spatial position. This representation of motion segmentation through tunnels can be employed to obtain efficient motion predictions for video coding applications.

All the stages of the proposed approach are processed sequentially. Every stage is performed for the entire video before going to the next stage. Thus, the proposed motion segmentation method can not be used in online applications, without video partitioning.

III. Estimation of Particle Trajectories

In this section, we show how the particles are selected in each video frame, and how they are located in the video frames of a video sequence. The approach proposed by Sand and Teller [34] is used here, with a few modifications.

Some important properties of this trajectory estimation approach are outlined next. First, the estimation of particle trajectories does not require any temporal smoothness constraints. This means that it is robust to abrupt camera motions, and objects motion discontinuities. Second, the particle sampling density is adaptive, in the sense that regions with more details are sampled with more particles, while homogeneous regions are sampled with less particles. Thus, higher motion segmentation precision can be obtained in regions with more motion information, where higher density is necessary, while saving computation in regions with less motion information. Third, motion information can be inferred from neighboring particles, reducing the effect of the aperture problem in homogeneous regions, where there is not enough motion information. These properties suggest that this particle video approach potentially can estimate long-range coherent motion patterns, while representing details of regional motion by adapting the granularity locally. These properties are advantageous in video coding, since data redundancy can be explored in sets of frames that are not immediate neighbors, not only in pairs of adjacent frames, as in many other approaches [28], [35], [36].

A particle is created in a frame pixel when a maximum proximity criterion is satisfied (as discussed in [34]). As soon as a particle is created, it is tracked along subsequent frames, until the point it represents becomes occluded, or its location surpasses the visible frame area. In this work, addition and tracking of particles are performed in one pass over the video sequence. As suggested by Sand and Teller [34], more passes can be executed in different directions (forward and backward) aiming at a better distribution and localization of particles, at the cost of higher computational effort. However, we did not notice a considerable advantage in the final motion segmentation results by performing several passes.

The method for estimating particle trajectories used in this work can be divided in four steps: particle addition, particle propagation, particle pruning and particle location optimization. In the particle addition step, new particles are created and inserted in each frame of the video sequence, in pixel locations where a particle proximity criterion is satisfied (see [34]). This proximity criterion is adaptive to the video scene contents. In the particle propagation step, the positions occupied by particles in the frame at time t are propagated to the frame at time $t+1$ using particle motion information, which is obtained from the computation of optical flow vectors (see [34]). In this work, we use the optical flow method proposed by Sand and Teller [34]. In the particle pruning step, we eliminate particles that, after the execution of the propagation step, become occluded or leave the field of view. In the particle location optimization step, the positions of particles are re-adjusted by minimizing an energy function provided by Sand and Teller. These re-adjustments of the particle positions are used for correcting particle motion estimation errors that may occur in the particle propagation step, avoiding the propagation of such errors to the subsequent frames (see [34]). The algorithm initially adds particles to the first frame of the sequence. After that, the second frame in the sequence is processed, and the algorithm executes the particle propagation, pruning and location optimization steps for the second frame. Next, some particles are added to the second frame to satisfy the particle proximity criterion, and the process continues with particle propagation, pruning and location optimization, considering the positions of particles in the third frame. These steps are executed in the above mentioned order for the subsequent frames, until the end of the video is reached. The particle addition, propagation, and location optimization steps used in this work are the same as those proposed by Sand and Teller [34]. The next section shows the particle pruning method employed in our work. This method introduces a modification in the method proposed by Sand and Teller, so the number of false occlusions can be reduced. We observed experimentally that a few particle trajectory detection errors still may occur, and the best segmentation results are obtained when particles have longer lifetimes. Particle trajectory errors are handled separately in this work.

A. Particle Pruning

In the work proposed by Sand and Teller [34], the same objective function that is minimized to optimize (i.e., fine-tune) the location of particles is employed as a measure of particle location reliability in the particle pruning stage. This objective function is composed by three terms, namely $E_p^{[c]}$, E_f and E_d as described below.

The first term $E_p^{[c]}$ represents the projection error of the particle p in the frame at time t with respect to the particle first appearance in a frame of the sequence

$$E_p^{[c]}(p,t)=\Psi([I^{[c]}(x_p(t),y_p(t),t)-I^{[c]}(x_p(t_p^0),y_p(t_p^0),t_p^0)]^2)$$

where $I^{[c]}$ denotes the color channel c (i.e., R,G,B) of the video sequence; $x_p(t)$ and $y_p(t)$ represent the spatial coordinates of particle p in the frame at time t; and t_p^0 represents the time of the first appearance of particle p in the sequence. The term $\Psi(a) = \sqrt{a+\epsilon}$ is the robust norm,[2] with $\epsilon = 0.0001$.

The second term E_f represents the difference between the actual displacement of the particle p in the frame at time t, and the corresponding estimate based on the local optical flow displacement (see the first equation shown at the bottom of the page), where $\bar{u}$ and $\bar{v}$ are the horizontal and vertical components of the optical flow vectors, respectively.

The third term E_d measures the relative motion between linked particles p and q in the frame at time t[3] (see the second equation shown at the bottom of the page), where $l_{pq}(t)$ is the weight of the link between particles p and q in the frame at time t. This weight is inversely proportional to the difference of optical flow vectors at the corresponding particle positions. See details in [34].

Grouping these three terms, we have the complete objective function

$$E(p,t) = \sum_c E_p^{[c]}(p,t) + \alpha_f E_f(p,t) + \alpha_d \sum_{q \in L_p(t)} E_d(p,q,t)$$

where $L_p(t)$ denotes the set of particles linked to the particle p in frame at time t; α_f and α_d are constants that provide a compromise among the terms, and were set to 5 and 10 in the particle location optimization step, respectively (α_f and α_d were set based on experiments). The performance of the trajectories estimation is not very sensitive to changes in these constants. So, the values fixed here for α_f and α_d may work for a large variety of sequences. We only need to be careful when setting the values for α_f, since the term E_f acts like a positional constraint; consequently, values for α_f that are too small could lead to significant errors with respect to the optical flow estimates.

In Sand and Teller [34], a particle p is pruned in a frame at time t when its energy function $E(p,t)$ is larger than a threshold. In our work, particles are pruned only if they become occluded, or if they leave the field of view. We define the field of view as the set of points that are distant at least five pixels from the image boundaries. So, when a particle leaves the field of view area after the propagation stage in a given frame, it is removed from that frame. This avoids propagating incorrect optical flow estimates in regions where these estimates are unreliable.

To detect when a particle becomes occluded, we use the occlusion detection approach employed in the optical flow method proposed by Sand and Teller [34]. In order to determine the current frame regions that will become occluded in the subsequent frame, the flow field divergence and pixel projection error are combined as explained next. Let us define the optical flow divergence as

$$r_{div}(x,y,t) = \frac{\partial}{\partial x}\bar{u}(x,y,t) + \frac{\partial}{\partial y}\bar{v}(x,y,t)$$

where $\bar{u}(x,y,t)$ and $\bar{v}(x,y,t)$ are the horizontal and the vertical components of the optical flow vectors, respectively, at the pixel position (x,y) of the frame at time t. The divergence is positive for disocclusion boundaries, negative for occlusion boundaries and near zero for shearing boundaries. Since we are interested only in occlusion boundaries, the function $r_d(x,y,t)$ is defined

$$r_d(x,y,t) = \begin{cases} r_{div}(x,y,t), & r_{div}(x,y,t) < 0 \\ 0, & r_{div}(x,y,t) \geq 0. \end{cases}$$

The projection error $r_e(x,y,t)$ also provides an estimate of occlusion likelihood

$$r_e(x,y,t) = I(x,y,t) - I(x+\bar{u}(x,y,t), y+\bar{v}(x,y,t), t+1).$$

These two estimates are combined to obtain the occlusion mask, which is the set of locations where $r(x,y,t) = 1$, and $r(x,y,t)$ is given by [34]

$$r(x,y,t) = \begin{cases} 0, & e^{-\left(|r_d(x,y,t)|/2\sigma_d^2 + r_e(x,y,t)/2\sigma_e^2\right)} \geq \tau_r \\ 1, & e^{-\left(|r_d(x,y,t)|/2\sigma_d^2 + r_e(x,y,t)/2\sigma_e^2\right)} < \tau_r. \end{cases} \quad (1)$$

The values of σ_d, σ_e and τ_r were chosen as 0.3, 20, and 0.5, respectively, based on experiments. The performance of particle pruning is not considerably sensitive to the values of σ_d and σ_e. However, close attention should be paid when setting τ_r. Too small τ_r values allow particles representing occluded points of the scene to survive longer, possibly changing the

[2]This function is a differentiable form of the absolute value function and does not respond as strongly to outliers as the L^2 norm [34].

[3]In this work, linked particles are defined as pairs of particles satisfying adjacency constraints imposed by the Delaunay triangulation.

$$\begin{aligned} E_f(p,t) = {} & \Psi([\bar{u}(x_p(t-1), y_p(t-1), t-1) - (x_p(t) - x_p(t-1))]^2 \\ & + [\bar{v}(x_p(t-1), y_p(t-1), t-1) - (y_p(t) - y_p(t-1))]^2) \end{aligned}$$

$$\begin{aligned} E_d(p,q,t) = {} & l_{pq}(t)\Psi([(x_p(t) - x_p(t-1)) - (x_q(t) - x_q(t-1))]^2 \\ & + [(y_p(t) - y_p(t-1)) - (y_q(t) - y_q(t-1))]^2) \end{aligned}$$

occluded object representation. On the other hand, too large τ_r values may prune consistent particles prematurely. Fortunately, with our particle segmentation approach, a small portion of tracking errors—particles that are associated to one object but erroneously change to another object—can be recovered in the postprocessing stages (see Sections IV-B and IV-C) without degenerating the segmentation results considerably.

In this work, if a particle becomes occluded or leaves the visible field, this particle is eliminated and it is no longer considered in the tracking process, even if the image location currently associated with that particle position becomes disoccluded later.

IV. Segmentation of Particle Trajectories

The representation of motion in videos based on particle trajectories is flexible, allowing the identification of object motion without global motion constraints (e.g., camera model, rigid scene model, etc.), handling occlusions and providing adaptive local granularity. However, extracting high-level information about the moving structures based on particle trajectories is not a trivial task, for the reasons discussed next. Different particles that represent scene points of the same moving structure can have different lifetimes. For example, it may happen that a pair of particles in coherent motion along the video sequence does not coexist in the same video frame, because of their different lifetimes. Moreover, it is difficult to avoid the incidence of erroneous motion patterns, which may occur due to the noise, artifacts caused by illumination and/or quantization/compression, lack of motion information in parts of the scene, and the absence of a validation model for particle motion.

Only particles that appear simultaneously in at least two consecutive frames can be compared directly in terms of their motion. Nevertheless, two particles that do not coexist in any video frame can be compared indirectly by investigating how they move in relation to other particles that have lifetime intersections with these two particles. For example, let F_1 and F_2 be the sets of frames where two particles p_1 and p_2 exist but have no lifetime intersection (i.e., $F_1 \cap F_2 = \emptyset$). Other particle p_3 may have its lifetime represented by the set of frames F_3, such as $F_1 \cap F_3 \neq \emptyset$ and $F_2 \cap F_3 \neq \emptyset$. Thus, motion analysis can be performed by combining information of particle sub-sets. This idea of combining motion information (motion patterns) of data sub-sets is similar to the combination of data partitions in ensemble clustering [37]–[39].

According to the ensemble clustering philosophy, given a data set with n samples, there are different ways of generating clusters from this data set [38].

- Applying different clustering algorithms to the whole set of n samples.
- Applying the same clustering algorithm with different parameters.
- Applying clustering algorithms to different data representations (for example, using different feature spaces).
- Applying clustering algorithms to different data partitions (i.e., to sub-sets of the n samples).

The last item describes the approach adopted in this work for grouping particles in coherent motion. A clustering algorithm is applied to each set of particles that coexist in the video sequence (i.e., have lifetime intersection), and then these particle sub-set clusterings are combined in larger particle clusters.

The majority of ensemble clustering methods require the computation of pairwise similarities for all objects in the collection. In our case, the number of particles in a video can be very large, and the cost of computing a pairwise similarity matrix can be impracticable. An interesting option is to consider the integration of different data partitions as a cluster correspondence problem [38], where similar cluster groups are identified and combined, forming meta-clusters. An algorithm for combining several particle clusterings in meta-clusters will be presented in Section IV-A, using the ensemble clustering approach proposed in [38], modified for selecting automatically the number of meta-clusters.

Inconsistencies in the particle clustering process are expected, since errors may occur when tracking a large number of stochastically moving particles. For this reason, the clustering stage is followed by a cluster validation stage (see Section IV-B), where particle context, motion and spatial location are combined to check for possible inconsistencies. Finally, a filtering stage (see Section IV-C) is executed to eliminate outliers and small groups of particles that are not considered significant.

A. Ensemble Clustering of Particles

In order to group particles that are in coherent motion along the video sequence, we first identify the subsets of particles that present similar motion in neighboring frames. Let $P = p_1, p_2, \ldots, p_{n_p^v}$ be the whole set of n_p^v particles in the video sequence, and $\vec{e_p}(t, t+l)$ be the displacement vector of particle p between frames at time t and time $t+l$

$$\vec{e_p}(t, t+l) = \begin{bmatrix} x_p(t+l) - x_p(t) \\ y_p(t+l) - y_p(t) \end{bmatrix}.$$

For each frame at time t, three clusterings are computed with the particles present in this frame (i.e., $l = 1, 2, 3$), based on the following features: 1) displacement vectors between the adjacent frames ($\vec{e_p}(t, t+1)$); 2) displacement vectors between frames distant two time units ($\vec{e_p}(t, t+2)$); 3) displacement vectors between frames distant three time units ($\vec{e_p}(t, t+3)$).

These three clustering are used in each frame to reinforce the tendency of particles with similar motion patterns to group together, reducing the influence of outliers and inconsistencies in individual clusterings in the final partition.

Assuming n_t frames in the video sequence, we will have $n_h = 3n_t - 6$ clusterings of the set of particles in those n_t frames. Note that some displacement vectors $\vec{e_p}(t, t+l)$ can not be calculated: (a) $\vec{e_p}(t, t+1)$, $\vec{e_p}(t, t+2)$ and $\vec{e_p}(t, t+3)$, when $t = n_t$; (b) $\vec{e_p}(t, t+2)$ and $\vec{e_p}(t, t+3)$ when $t = n_t - 1$; and (c) $\vec{e_p}(t, t+3)$ when $t = n_t - 2$. Each clustering $\{H^{(j)} | j \in \{1, \ldots, n_h\}\}$ divides the set of n_p^v particles in $n^{(j)}$ clusters $\{h_i^{(j)} | i \in \{1, \ldots, n^{(j)}\}\}$, $H^{(j)} = \{h_i^{(j)}\}$, where $h_i^{(j)}$ represents the ith cluster of the jth clustering, and $n^{(j)}$ denotes the number of clusters of the jth clustering.

The mean-shift method [40] is employed in this work to obtain the clusters $h_i^{(j)}$ in $H^{(j)}$, because of the ability of the mean-shift method to identify clusters with arbitrary shapes in feature space. The bandwidth of the mean-shift kernel was set to $3 \cdot l$, for

TABLE I
EXAMPLE OF A HYPER-GRAPH WITH 8 HYPER-EDGES

	$H^{(1)}$			$H^{(2)}$			$H^{(3)}$	
	$h_1^{(1)}$	$h_2^{(1)}$	$h_3^{(1)}$	$h_1^{(2)}$	$h_2^{(2)}$	$h_3^{(2)}$	$h_1^{(3)}$	$h_2^{(3)}$
p_1	1	0	0	1	0	0	1	0
p_2	1	0	0	0	1	0	0	0
p_3	0	1	0	0	1	0	0	1
p_4	0	1	0	0	1	0	0	1
p_5	0	0	1	0	0	1	0	0
p_6	0	0	1	0	0	0	0	0

all the frames $(t, t+l)$ and $l = 1, 2, 3$. The bandwidth increases with l in order to reduce clustering diversity and, consequently, to produce meaningful cluster correspondences, because low diversity within clusters is required by the ensemble clustering algorithm employed in this work. Note that the bandwidth of the mean-shift kernel is related to the expected range of relative motions between objects (i.e., particle clusters). The bandwidth can be tuned according to the kind of object motion (slow, fast) found in the sequence. Nevertheless, it is not affected by global motion, such as the translational motion induced by a moving camera, for example.

At this stage, we have three clusterings $\{H^{(j)}\}$ for each frame. Only one data partition is obtained for the whole set of particles in the video sequence, by combining all clusterings (three for each frame) in a single clustering H. To perform this task, we use the meta-clustering algorithm (MCLA) [38], as detailed next.

Each particle $p_i (i = 1, 2, \ldots, n_p^v)$ is assigned to one of the clusters $h_i^{(j)}$ in each clustering $H^{(j)}$ based on its motion patterns in different time spans $(l = 1, 2, 3)$. Therefore, each cluster $h_i^{(j)}$ in clustering $H^{(j)}$ can be represented by a binary label vector, where a particle p_i is labeled as "1" if it belongs to $h_i^{(j)}$, or "0" otherwise (see Table I).

Initially, according to the MCLA approach, the n_h clusters $h_i^{(j)}$, represented by binary label vectors, are transformed in a hyper-graph. A hyper-graph is constituted by nodes and hyper-edges, and it is a generalization of a graph in the sense that a hyper-edge can connect any set of nodes (while in a regular graph, an edge connects exactly two nodes). Each clustering $H^{(j)}$ is represented by a binary matrix, with n_p^v rows (one for each particle of the entire video sequence) and $n^{(j)}$ columns (i.e., a concatenation of $n^{(j)}$ binary label vectors, one for each cluster). In our hyper-graph representation, nodes represent particles and the hyper-edges represent clusters of particles that jointly occur in 1, 2, and 3 consecutive frames. Table I shows a simple example of a hyper-graph. In this example of a simple sequence of 3 frames, the lines of the Table represent the particles ($p_{\{1,\ldots,6\}}$), the clusterings are represented by the binary matrices $H^{(1,\ldots,3)}$, and the individual clusters are represented by the hyper-edges $h_{1,2,3}^{(1)}$, $h_{1,2,3}^{(2)}$, and $h_{1,2}^{(3)}$. Note that, since we have three frames in this example, three clusterings are formed using: 1) motion vectors between frames 1 and 2 ($H^{(1)}$); 2) motion vectors between frames 1 and 3 ($H^{(2)}$), and 3) motion vectors between frames 2 and 3 ($H^{(3)}$). Since a particle can belong to only one cluster $h_i^{(j)}$ in each clustering $H^{(j)}$, the lines

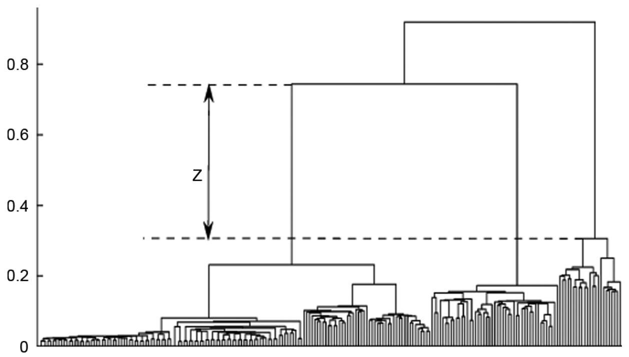


Fig. 1. Example of a dendrogram of meta-clusters. Z represents the longest range of consecutive threshold values in which the number of meta-clusters does not change $(K = 3)$.

in the corresponding binary matrix sum up to "1" when the particle cluster is known (i.e., the particle is defined in the frame interval $[t, t+l]$ used to produce the clustering $H^{(j)}$). Lines corresponding to particles that were not assigned to any cluster sum up to "0" (i.e., the corresponding particles do not exist in the frame interval $[t, t+l]$ over which the clustering $H^{(j)}$ is defined).

The concatenated matrix $\mathbf{H} = H^{(1,\ldots,n_h)} = (H^{(1)} \ldots H^{(n_h)})$ is a hyper-graph, with n_p^v nodes (i.e., particles) and $n_{all} = \sum_{j=1}^{n_h} n^{(j)}$ hyper-edges (i.e., particle clusters). Each $H^{(j)}$ column vector $h_i, i = 1, \ldots, n_{all}$, is a hyper-edge of the hyper-graph $\mathbf{H}$, and represents one of the particle clusters of the set of clusterings $H^{(j)}, j = 1, \ldots, n_h$. The MCLA approach groups particle clusters (i.e., hyper-edges) in meta-clusters based on their similarity measured by the Jaccard distance [see (2)]. The final particle segmentation is obtained by assigning each particle in the sequence to one meta-cluster, and these meta-clusters will eventually lead to the final segmentation, as explained in Section V.

The meta-clusters are obtained by hierarchical clustering of hyper-edges. To do that, a symmetric similarity matrix m is computed, where $m(h_s, h_t) = m(h_t, h_s)$, containing the calculated similarity between each pair of hyper-edges h_s and h_t of the hyper-graph $\mathbf{H}$. The similarity between pairs of clusters (i.e., hyper-edges) h_s and h_t is measured by the Jaccard distance

$$m(h_s, h_t) = \frac{h_s' \cdot h_t}{\|h_s\|_2^2 + \|h_t\|_2^2 - h_s' \cdot h_t}. \tag{2}$$

The computed similarity matrix m is then used as an input of the hierarchical clustering algorithm, namely the single link method [41]. The ideal number K of meta-clusters is selected as the number of clusters that is more stable in the hierarchical clustering dendrogram. We observe how the number of clusters vary as the threshold values increases from "0" to "1", when the dendrogram is built; K is the number of detected clusters in the longest range of consecutive threshold increases when the dendrogram is built, for which the number of clusters does not change. This is illustrated in Fig. 1, where the range Z in the dendrogram defines the ideal number of meta-clusters $K = 3$ (since this is the longest sequence of threshold value changes for which the number of meta-clusters is constant).

TABLE II
EXAMPLE OF 3 META-CLUSTERS, REPRESENTED BY THEIR RESPECTIVE META-HYPER-EDGES

	M_1	M_2	M_3
p_1	0	1	0
p_2	0	0.33	0.33
p_3	0	0	1
p_4	0	0	1
p_5	1	0	0
p_6	0.5	0	0

After the K meta-clusters is identified, all the hyper-edges belonging to the same meta-cluster are grouped together in a single meta-hyper-edge. Suppose we consider the original logical values of particles in the hyper-edges as being weights indicating the association of particles to the corresponding clusters h_i, and also to the clusterings $H^{(j)}$ ("1" indicating a strong association and "0" indicating no association). Then, particles are assigned to meta-clusters based on their weights (or association to the meta-cluster), and these weights are computed by the particle weight average in all hyper-edges h_i belonging to each meta-cluster. Supose we have identified three meta-clusters $M_{1,\ldots,3}$ in our hyper-graph example shown in Table I, each one of them composed by the following hyper-edges: 1) M_1: $h_3^{(1)}$ and $h_3^{(2)}$; 2) M_2: $h_1^{(1)}$, $h_1^{(2)}$ and $h_1^{(3)}$; 3) M_3: $h_2^{(1)}$, $h_2^{(2)}$ and $h_2^{(3)}$. The meta-hyper-edges corresponding to these three meta-clusters are shown in Table II. Recall that a particle can be associated to different clusters h_i, and these clusters can be associated to different meta-clusters (for example, in some frames a particle is moving in relation to the background, but the same particle can be static in other frames). Calculating the average particle weight in the hyper-edges will result in a meta-hyper-edge that has an entry for each particle in the video sequence, describing the degree of association of this particle with the corresponding meta-cluster. Stronger this association, closer to "1" is this entry; while weaker associations have values closer to "0".

The last step in the ensemble clustering is the assignment of each particle to the meta-cluster to which it has the larger association degree (i.e., to the meta-hyper-edge vector that has the largest calculated particle average weight among all meta-hyper-edges), and ties are broken randomly. In our example (Table II), we would have the following assignments: 1) $p_1 \rightarrow M_2$; 2) $p_2 \rightarrow$ assigned to either M_2 or M_3, at random; 3) $p_3 \rightarrow M_3$; 4) $p_4 \rightarrow M_3$; 5) $p_5 \rightarrow M_1$; and 6) $p_6 \rightarrow M_1$. At the end of the particle assignment process to the K meta-clusters, the final set of particle meta-clusters $\{M_b | b \leq K\}$ is obtained. The final number of meta-clusters can be smaller than K because some meta-clusters can have no particle assigned. Thus, there will be b labels at the end of the particle clustering process (a maximum of K).

Although this approach uses clusterings obtained from pairs of frames, the ensemble clustering tends to extract long-range patterns, without suffering from the error accumulation problem that is present in the purely frame-by-frame sequential approaches [36], [42]. Furthermore, new objects that appear along the video do not need special treatment.

B. Particle Meta-Clustering Validation

As mentioned before, particle tracking errors may occur, resulting in incorrect particle-to-meta-cluster assignments. To detect these inconsistencies after the ensemble clustering stage (Section IV-A), a cluster validation step is performed by analyzing trajectories in the context of particles grouped together (same meta-cluster). This is particularly important when particles are assigned to one object but migrate to another object during the tracking process, because of occlusion detection imperfections.

Let $p = (x_p(t), y_p(t))$ be the pth particle, represented by its spatial coordinates in frame at time t. The context of this particle is defined by a window of size $2W_V + 1$, given by $(x_p(t) + \Delta_{V_x}, y_p(t) + \Delta_{V_y})$, where

$$-W_V \leq \Delta_{V_x}, \Delta_{V_y} \leq W_V, \quad \Delta_{V_x}, \Delta_{V_y} \in \mathbb{Z}.$$

In all our experiments, a context of size $W_V = 2$ was used.

The context of a particle moving coherently with its neighboring particles often does not change from one frame to another, since the set is moving coherently and we can assume motion approximately translational between adjacent frames. The projection error of a point in a window $[-W_V, W_V]$ at sub-pixel level, that specifies the context of particle p, given the motion of the particle between frame at time t and frame at time $t - \rho_V$, can be computed as follows (see the equation shown at the bottom of the page). In order to detect context changes, we compute the mean of the w-best matches (i.e., the mean of the w smallest motion estimation errors of $D^{[\Delta_{V_x}, \Delta_{V_y}]}(p, t, \rho_V)$ in $-W_V \leq \Delta_{V_x}, \Delta_{V_y} \leq W_V$), and if this mean value is large enough (i.e., larger than a threshold τ_{best}, with $\tau_{\text{best}} = 100$), we assume that particle p changed its context in the frame at time t, and is marked as inconsistent. Based on experiments, we verified that $w = 15$ and $\rho_V = 3$ offer a good compromise between false positives and false negatives. Note that the choice for ρ_V is strongly related to the bandwidth of the mean-shift kernel. Both constants must be defined in terms of the expected range of relative motion between objects. The slower the motion is, the larger is ρ_V (and smaller the bandwidth kernel is); however, the faster is the motion, the smaller ρ_V is (and larger the bandwidth kernel is). The choice of a value for w is discussed below.

$$D^{[\Delta_{V_x}, \Delta_{V_y}]}(p, t, \rho_V) = \left\| I(x_p(t) + \Delta_{V_x}, y_p(t) + \Delta_{V_y}, t) - I(x_p(t - \rho_V) + \Delta_{V_x}, y_p(t - \rho_V) + \Delta_{V_y}, t - \rho_V) \right\|_2$$

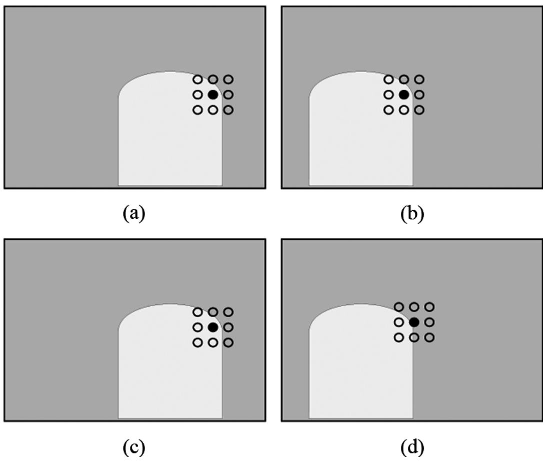


Fig. 2. Context change detection: (a)–(b) w-best matches $< \tau_{\text{best}}$; (c)–(d) w-best matches$\geq \tau_{\text{best}}$ (a) frame at time $t - \rho V$ (b) frame at time t (c) frame at time $t - \rho V$ (d) frame at time t.

We use only the w-best matches in the window context to reduce the influence of particles near motion boundaries, as their contexts are affected by different motions in adjacent regions. However, if w is too small, slow transitions between adjacent regions would not be detected as context changes. An example of context change in a 3×3 window is illustrated in Fig. 2. Note that, with a small w, a particle context change could not be detected if the transition is slow enough to obtain a few good matches.

Let $\Omega_p = \{\omega_p^1, \omega_p^2, \ldots, \omega_p^{n_\omega}\}$ be the ordered set of indices of the frames where possible context changes occur for particle p. The lifetime of particle p is then divided in intervals

$$\left\{[1, \omega_p^1), [\omega_p^1, \omega_p^2), \ldots, [\omega_p^{n_\omega - 1}, \omega_p^{n_\omega}), [\omega_p^{n_\omega}, n_t]\right\}. \qquad (3)$$

Recall that a particle must move consistently with the collective motion of the other particles in its meta-cluster M_b, and its spatial vicinity should not change much. To determine if a context change actually occurred (and the particle trajectory has been incorrectly calculated), we follow the motion and the spatial location of particle p, comparing it with meta-cluster prototypes (as explained below) in the intervals shown in (3). Thus, the similarity between the motion patterns of particle p and the prototype of each meta-cluster M_b is given by

$$S(p, b, t) = e^{-(D_m(p,b,t)/2\cdot\sigma_m^2 + D_s(p,b,t)/2\cdot\sigma_s^2)} \qquad (4)$$

where $D_m(p, b, t)$ and $D_s(p, b, t)$ denote the motion and spatial differences, respectively, between the particle p and the prototype of the meta-cluster M_b in frame at time t, which are defined below. The standard deviations σ_m and σ_s were set to "1". The motion difference $D_m(p, b, t)$ is defined as

$$D_m(p, b, t) = \sqrt{(u_p(t) - u^{[b]}(t))^2 + (v_p(t) - v^{[b]}(t))^2}$$

where $u_p(t) = x_p(t) - x_p(t-1)$, $v_p(t) = y_p(t) - y_p(t-1)$, and $u^{[b]}(t)$ and $v^{[b]}(t)$ are, respectively, the horizontal and vertical components of the meta-cluster M_b representative motion vector between the frames at time t and time $t-1$. This representative motion vector is computed as follows.

1) Compute the set of motion vectors $\{[u_p(t), v_p(t)] | \forall p \subset M_b\}$.
2) Compute the reduced order of the respective motion vectors, in relation to a reference vector given by the minimum values of horizontal and vertical particles displacements among all particles belonging to meta-cluster M_b: $[\min(u_p(t) | \forall p \subset M_b), \min(v_q(t) | \forall q \subset M_b)]$.
3) The vector corresponding to the median in the sorted distances $\|\cdot\|_2$ to the reference vector is assigned to $[u^{[b]}(t), v^{[b]}(t)]$.

The spatial difference to particle p, $D_s(p, b, t)$, is defined as

$$D_s(p, b, t)1 = \frac{1}{k} \cdot \sum_{q=1}^{k} \sqrt{(x_p(t) - x_{p,q}^{[b]}(t))^2 + (y_p(t) - y_{p,q}^{[b]}(t))^2}$$

where $\{(x_{p,q}^{[b]}(t), y_{p,q}^{[b]}(t)) | q = 1, \ldots, k\}$ are the spatial coordinates of the k nearest particles that belong to the meta-cluster M_b in relation to the particle p in frame at time t.

In order to determine to which meta-cluster the particle p should be assigned in the interval $[\omega_p^1, \omega_p^2)$, we verify which meta-cluster M_b maximizes the following similarity measure [see (4)] in this interval

$$b_{\max} = \max_b \sum_{t=\omega_p^1}^{\omega_p^2 - 1} S(p, b, t). \qquad (5)$$

Note that if a context change is detected, it does not necessarily imply a meta-cluster change after validation. It just yields a marker that give us evidence that the particle trajectory estimation can be incorrect. The analysis of spatial and motion coherence expressed in (5) indicates if a meta-cluster change has occurred.

C. Spatial Filtering

The last stage in the classification process is the particle spatial filtering. The goal of spatial filtering is to eliminate outliers and groups of adjacent particles that are not significant (i.e., assigned to small isolated region fragments).

To represent the particles spatial adjacency, the Delaunay triangulation $DT(t)$ is computed based on the particle positions $(x_p(t), y_p(t))$ in each frame at time t. Two particles are considered adjacent if they share an edge in the triangulation $DT(t)$. The association between adjacent particles is represented by assigning binary weights to the edges of $DT(t)$; that is, an edge receives "1" if it connects two particles belonging to the same meta-cluster, or it receives "0" if connects particles belonging to different meta-clusters. All the connected components of $DT(t)$ are examined[4] and very small ones (less than 20 particles) are assigned to other meta-clusters. These components are assigned to the meta-cluster that shares more edges with weight "0", i.e., to the other meta-cluster with which it shares the largest spatial border.

V. Dense Segmentation Extraction

In many computer vision and image processing tasks, and in many video coding problems, it is necessary to extract a dense

[4]Two particles are in the same connected component if and only if there is a path between them composed only by edges of weight "1".

representation of motion segmentation. It means that we must determine which pixels are assigned to each moving object.

At this stage, we know the particles positions in each frame, as well as the particles meta-cluster labels. Then, we only need to assign each pixel to the most similar particle meta-cluster. To perform this task, we compare pixels with sets of particles in terms of motion and spatial proximity using implicit functions, as explained next. Let $P = p_1, p_2, \ldots, p_{n_p^v}$ be the whole set of n_p^v particles of the video, and $\lambda_p \in \{1, \ldots, K\}$ the set of labels that indicates for each particle, to which meta-cluster $b = 1, \ldots, K$ the particle is associated with. So, to each particle p in a frame at time t, represented by its spatial coordinates (x_p^t, y_p^t), is assigned a multivariate gaussian kernel [see (6), shown at the bottom of the page], where $\hat{u}_p^t = x_p^t - x_p^{t-1}$ and $\hat{v}_p^t = y_p^t - y_p^{t-1}$ represent the displacement of particle p in frame at time t, and Σ is the covariance matrix given by

$$\Sigma = \begin{bmatrix} \sigma_S & 0 & 0 & 0 \\ 0 & \sigma_S & 0 & 0 \\ 0 & 0 & \sigma_M & 0 \\ 0 & 0 & 0 & \sigma_M \end{bmatrix}$$

where, σ_S and σ_M are the spatial and motion standard deviations, set to 50 and 1, respectively. These coefficients are chosen based on a compromise between precision and smoothness of the objects boundaries, and can be modified according to the application. The smaller the value of σ_S, the more precise the boundaries will be. The larger the σ_S value, the smoother the boundaries will be. On the other hand, σ_M controls the weight of motion information used in pixel classification. Large σ_M values should be avoided because it would cause fragmentation of object regions. It means that each particle will be associated to an implicit function, that defines the likelihood of assigning a pixel (x, y) to the motion pattern represented by particle p in a 4-D space (two spatial dimensions and two motion dimensions). Thus, pixels located near the particle p, with motion similar to particle p, will yield large $G_p^t(x, y, \bar{u}, \bar{v}, t)$ values, while pixels far away from the particle p, or with distinct motion vectors, will yield small $G_p^t(x, y, \bar{u}, \bar{v}, t)$ values. In order to assign one pixel to a meta-cluster, we compare it with all the particles of each meta-cluster using (6), and assign the pixel to the particle meta-cluster which yields the largest sum of particle implicit functions.[5] A pixel of the image $I(x, y, t)$ is then assigned to

[5]Alternatively, we can compare a pixel only with the k-nearest particles. In our experiments, we compared each pixel with the 30 nearest particles, and obtained results virtually equal to those obtained by comparing pixels with all the particles, with the advantage of reducing the dense segmentation extraction computation time in more than 90%.

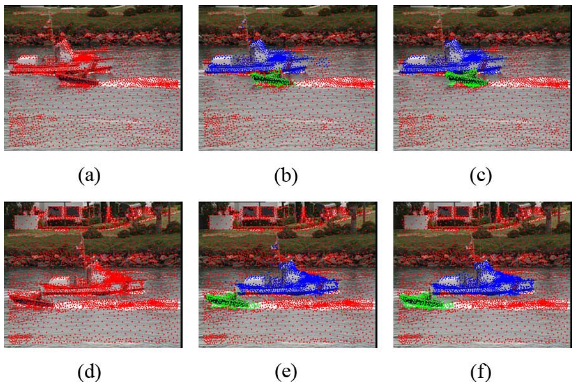


Fig. 3. Particle segmentation for the frame 5 (first row) and frame 45 (second row) of the *coastguard* sequence: (a), (d) particle tracking results; (b), (e) particle meta-clustering results; and (c), (f) final particle segmentation.

the particle meta-cluster M_N that maximizes the sum of the Gaussian kernels at the corresponding pixel position

$$\max_N \sum_{\{p|\lambda_p = N\}} G_p^t(x, y, \bar{u}(x, y, t), \bar{v}(x, y, t), t). \tag{7}$$

Note that we use the optical flow components ($\bar{u}(x, y, t)$ and $\bar{v}(x, y, t)$) as the motion information at pixel level, and compare them with the motion of particles $(\hat{u}_p^t, \hat{v}_p^t)$ using the implicit functions given by the Gaussian kernels sum. The pixel is then assigned to the meta-cluster containing particles near the pixel that are moving more coherently with respect to the local optical flow, i.e., the maximum of (7). The more accurate the optical flow estimate, the more accurate the segmented object motion contours. Using sum of Gaussian kernels results in spatially smooth contours, even when few particles are available near motion object boundaries.

The final result of the pixel assignments to the particle meta-clusters in all frames of the sequence can be seen as a set of volumes (representing objects) in the spatio-temporal domain, where pixels are represented by voxels.

VI. Experimental Results

In this section, we present some experimental results to illustrate the performance of our proposed approach. Fig. 3 shows results of the proposed segmentation method for the fifth and the 45th frames of the classic *coastguard* sequence in the first and second rows, respectively. The whole tested sequence has 50 frames, each frame with 288×352 pixels. In (a) and (d), we show particle tracking results for these frames, and the 5427 particles used in the whole sequence. In (b) and (e), the three meta-clusters obtained with the proposed motion segmentation

$$G_p^t(x, y, u, v, t) = \frac{1}{(2\pi)^2 |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} \begin{bmatrix} x - x_p^t \\ y - y_p^t \\ \bar{u}(x, y, t) - \hat{u}_p^t \\ \bar{v}(x, y, t) - \hat{v}_p^t \end{bmatrix}^{\mathrm{T}} \Sigma^{-1} \begin{bmatrix} x - x_p^t \\ y - y_p^t \\ \bar{u}(x, y, t) - \hat{u}_p^t \\ \bar{v}(x, y, t) - \hat{v}_p^t \end{bmatrix} \right) \tag{6}$$

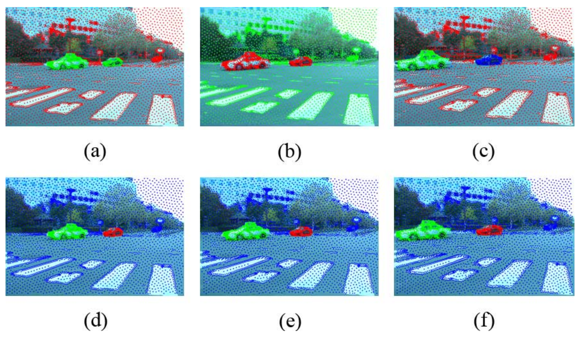


Fig. 4. Particle segmentation for the frame 5 (first column), frame 15 (second column) and frame 25 (third column) of the *cars* sequence: (a)–(c) individual clustering results and (d)–(f) the final segmentation results for the correspondent frames.

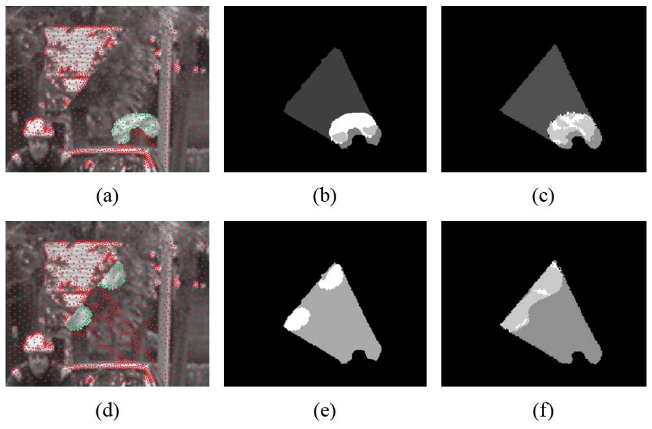


Fig. 5. Segmentation for the frame 5 (first row) and frame 30 (second row) of a synthetic video: (a), (d) particle segmentation results; (b), (e) object tunnel, occlusion volumes and exposed volumes obtained with the proposed method; and (c), (f) object tunnel, occlusion volumes and exposed volumes for the method proposed by Ristivojevi e Konrad [10].

method are presented. The meta-clusters are represented by particles of distinct colors (red, green, and blue). The proposed method segmented correctly the sequence in three objects moving distinctly, in spite of the high noise contamination level in the original sequence, and the water surface fluctuations. The final results of the segmentation process—after meta-cluster validation and spatial filtering—are shown in (c) and (f). Note that some particles belonging to the blue meta-cluster, that had been wrongly classified to the background, were correctly re-classified after meta-cluster validation and spatial filtering.

Individual clusters, as well as the final segmentation for the video *cars*, used in the work of Sand and Teller [34], are shown in Fig. 4. The first row of Fig. 4 shows the results of individual meta-clusters (i.e., based only on the neighboring frames) obtained using the mean-shift algorithm applied to the frames 5, 15, and 25. Note that the perspective projection effect in this video is quite pronounced. For this reason, and due to the fact that mean-shift uses a fixed bandwidth, restricting the range of object motions detected, was not possible to distinguish between the two cars in the foreground in frames 5 and 15, using individual two-frames clustering. However, the ensemble clustering approach identifies them as distinct objects, as shown in the second row of Fig. 4 (final results). This occurs because the ensemble clustering recognizes groups of particles that tend to be clustered together along the entire sequence, returning meaningful results even when motion boundaries are unreliable or incorrect individual clusterings are found. This property is also important in video coding, as the data redundancy of individual objects is better explored, without breaking up object trajectories in several parts.

Fig. 5 presents results of the proposed method for a 30 frames synthetic video, used in the work of Ristivojevic and Konrad [10] for the extraction of object tunnels, occlusion volumes, and exposed volumes. The authors suggested that these concepts could be used in next-generation video compression methods. Thus, we compared the results obtained by Ristivojevic and Konrad with our proposed approach. This video consists of a synthetic beam-shaped object that moves against a static background. This object has its motion described by an affine transformation (with rotation, translation, and scale), and suffers occlusion, inducing a topological spatial change. The tenth and the 30th frames of the sequence are shown in Fig. 5 (first and second row, respectively). The corresponding particle segmentations are shown in (a) and (d). The dense segmentation results (see Section V), are shown in (b) and (e), as well as the corresponding occlusion volumes and exposed volumes, as proposed by Ristivojevi and Konrad [10]. The labels, from black to white, represent: 1) background; 2) background occlusion volume; 3) background exposed volume; 4) object; and 5) object occlusion volume. The corresponding results obtained by Ristivojevi and Konrad method are shown in (c) and (f). We obtained the occlusion volumes and exposed volumes by estimating an affine transformation for each region in each frame, and propagating these regions through the sequence using these transformations. This affine transformation is estimated based on the motion of the particles belonging to the same region. To compare the segmentation accuracy of both methods, we manually created a ground truth for this sequence, and obtained a rate of 99.75% correctly classified object pixels with our method, against 99.35% obtained by Ristivojevi and Konrad [10]. The accuracy measurements refer only to object pixel classification (object versus background, in this case). We can see that the proposed approach deals with occlusion and topological changes in a straightforward manner, and obtain occlusion volumes and exposed volumes reliably, without the limitations of the approach from Ristivojevic and Konrad (as discussed in Section I). Further, the results demonstrate that our approach deals in a straightforward way with topological changes caused by occlusions. Note that using the proposed segmentation approach, together with the dense segmentation, we also have the information of the corresponding particles that can be used as a motion cue in several applications. For example, occlusion and motion within the objects can be inferred through the information of particles and dense segmentation combined together. In the experiment showed above, we employed an affine transformation to perform this task. However, it should be noted that the segmentation algorithm is general and does not implies any motion constraint.

In Fig. 6, we present the results of our proposed method for a different synthetic video, showing background motion and three moving objects in the foreground. One object in the foreground

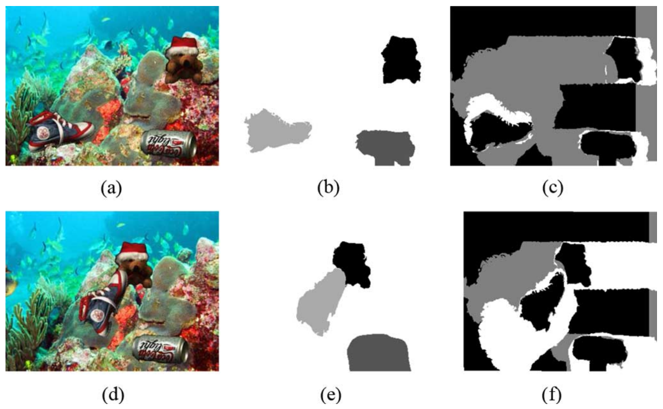


Fig. 6. Segmentation results for frame 5 (first row) and frame 20 (second row) of a synthetic video with three moving objects: (a), (d) original frames; (b), (e) object segmentation labels; (c), (f) object tunnels, occlusion volumes and exposed volumes.

rotates and translates, while the others only undergo translational motion. The original frames 5 and 20 of the sequence (which is composed by 30 frames) are shown in (a) and (d). The labels of the segmented objects are shown in (b) and (e), while object tunnels (black), occlusion volumes (gray) and exposed volumes (white) are shown in (c) and (f). The proposed method produced a correct object classification pixel rate of 96.12%. An interesting property of this video is that one object (can of soda), moves along with the background for ten frames, while exhibiting a different motion in the remaining frames of the sequence. Even without motion boundaries along one third of the sequence, the proposed method has segmented this object from the background in all frames, due to its ability of extract prolonged patterns.

In order to evaluate the ability of the proposed method to extract temporally redundant regions in videos, we computed temporal predictions along object tunnels. This was done by fitting an affine transformation for each tunnel and each frame, based on the motion of particles belonging to the corresponding tunnel in the corresponding frame. Then, this affine transformation was used to compensate tunnel motion to compute the pixel predictions in each frame. The entropy of the residual prediction errors is calculated and compared with the entropy of the original video data, and the entropy of block matching residual errors. The block matching approach used here for comparison is Adaptive Rood Pattern Search (ARPS) [43]. The results are shown in Table III, and indicate that our proposed method potentially can produce lower bit-rates than the block-matching approach. The PSNR of the block matching predictions and the intratunnel predictions are presented in Tables IV and V, respectively. The minimum, maximum and mean PSNR values for all frames in each video sequence are shown. However, these results should not be considered video coding results in any way, as we are not measuring real bitrates. They are just an indicative that the tunnels obtained though the proposed segmentation method are temporally very redundant, using the popular block matching approach as a reference. To employ the segmentation approach in a video coding framework, several other aspects should be considered, as the object shape coding, motion parameters, key-frames selection, amount of loss allowed, residual spatial coherence, source modeling, entropy coding, etc. We intend to use the proposed segmentation approach in a complete video coding framework in a future work.

All the experiments were conducted using a nonoptimized Matlab code, except for the optical flow, which was coded in C. Running the proposed approach in a PC-based computer with a 1.6-GHz dual core processor, for a sequence of 50 frames, with about 6000 particles (coastguard sequence), the computation took around 7 hours. The bottleneck of the segmentation performance is on the optical flow and particle tracking computations (both are variational methods), taking about 35% and 45% of the total computation time, respectively. Since the stages of particle tracking (see Section III) and particle segmentation (see Section IV) are computed only using the set of particles, and the cardinality of this set is much smaller than the cardinality of the set of pixels (i.e., there are fewer particles than pixels), we believe that with an optimized code and further reducing the number of particles utilized, the performance of these stages could be executed in real-time (or near real-time) using powerful machines. The computation of dense segmentation extraction can be optimized to run in approximately linear time, as discussed in Section V. Thus, in order to use this approach

TABLE III
ENTROPIES OF THE RESIDUAL PREDICTION ERRORS AND ORIGINAL DATA FOR VIDEO SEQUENCES

Video	Tunnel Prediction Errors	Block-Matching Residual Errors	Original Data
synt-objects	4.77	6.60	7.84
synt-seq	1.22	3.68	5.22
cars	4.32	5.47	7.22
coastguard	5.33	6.67	7.62
bus	5.44	7.27	7.31

TABLE IV
PSNR OF BLOCK MATCHING PREDICTIONS

Video	Mean PSNR	Min PSNR	Max PSNR
synt-objects	15.15 dB	14.74 dB	15.36 dB
synt-seq	19.94 dB	19.28 dB	20.54 dB
cars	18.52 dB	17.92 dB	19.11 dB
coastguard	17.31 dB	15.83 dB	19.20 dB
bus	14.02 dB	13.49 dB	14.53 dB

TABLE V
PSNR OF INTRA-TUNNEL PREDICTIONS

Video	Mean PSNR	Min PSNR	Max PSNR
synt-objects	22.66 dB	21.10 dB	23.81 dB
synt-seq	30.61 dB	28.68 dB	33.07 dB
cars	30.92 dB	28.10 dB	35.27 dB
coastguard	26.39 dB	22.23 dB	30.07 dB
bus	21.62 dB	19.98 dB	23.11 dB

in applications with real-time requirements, we suggest to substitute the variational optical flow employed here by a more efficient optical flow method. The ensemble clustering strategy employed in this work uses information about particle segmentation in every frame of the sequence (see Section IV-A). Consequently, the object segmentation for each frame of the sequence only can be known after all frames have been processed. Therefore, even if the previous steps (i.e., optical flow computation, particle trajectory estimation and mean-shift clustering) are computed in real-time, no frame segmentation is delivered until the sequence is entirely processed. An alternative to minimize the delay would be to divide the sequence in smaller parts, with a fixed or adaptive number of frames, using buffers to store information about previous frames.

VII. Conclusion

A method for unsupervised identification of coherent motion in adaptively sampled videos was proposed in this work. This technique provides a new way of linking low-level information in videos to high-level concepts that can be employed directly in video coding. This approach can be useful in many other image processing and computer vision tasks, including object tracking, information retrieval and video analysis.

The proposed method allows us to identify temporally discontinuous motion patterns, and is robust to abrupt camera motion. Besides, no global motion constraints are imposed to the scene and/or the objects. The aperture problem is reduced by inferring motion in homogeneous regions by the propagation of motion information in the neighborhood of the spatial samples (i.e., particles). However, the solution used to approach the aperture problem has a drawback: when homogeneous regions become occluded, motion propagation can yield erroneous motion estimates, which must be corrected by additional consistency steps, at the expense of more computational effort.

The proposed particle segmentation method uses ensemble clustering to combine particle clusters obtained for adjacent frames, allowing the identification of long-range motion patterns, which we represent as spatio-temporal volumes called tunnels. The identification of long-range motion patterns is crucial to take full advantage of temporal redundancy in segmentation-based video coding. The mean-shift algorithm [40] is employed to obtain the particle clusters associated to adjacent frames, and has the important property of identifying clusters with arbitrary shapes in feature space. However, the mean-shift algorithm presents an important drawback: it requires the use of a fixed bandwidth, reducing the magnitude and variety of detected motion patterns. Thus, by setting a fixed mean-shift bandwidth, we restrict the performance of the method to a certain range of motion magnitudes between objects. The value of ρ_V (see Section IV-B) also needs to be tuned according to the range of motion magnitudes between objects. This can be a drawback in applications where a wide range of motion magnitudes is expected. Another limitation of the proposed segmentation method concerns the type of motions that are better handled with this approach. Since we generate clusters based on similarity of the 2-D projected motion vectors, sequences with pronounced perspective effects and/or with complex 3-D spatial motion tend to be over-segmented.

Some experimental results obtained with the proposed motion segmentation method were presented, as well as PSNR values for temporal predictions along segmented tunnels, and the entropies of the prediction errors. Comparisons with block matching predictions indicate that the proposed segmentation approach potentially can be used in video coding with advantages in terms of bit-rate, PSNR and flexibility to handle moving objects.

Acknowledgment

The authors would like to thank Dr. P. Sand for providing the optical flow code used in [34] and the anonymous reviewers for providing valuable suggestions that have helped to improve this paper.

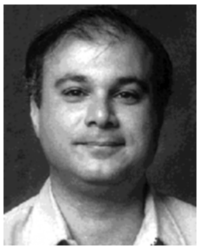
References

[1] T. Boult and L. G. Brown, "Factorization-based segmentation of motions," in *Proc. IEEE Workshop Visual Motion*, 1991, pp. 179–186.
[2] Costeira and Kanade, "A multibody factorization method for independently moving objects," *Int. J. Comput. Vis.*, vol. 29, pp. 159–179, 1998.
[3] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *Proc. 8th IEEE Int. Conf. Computer Vision*, 2001, vol. 2, pp. 586–591.
[4] A. Gruber and Y. Weiss, "Multibody factorization with uncertainty and missing data using the em algorithm," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 2004, vol. 1, pp. I-707–I-714.
[5] Yan and Pollefeys, "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate," presented at the ECCV, 2006.
[6] R. Vidal and R. Hartley, "Motion segmentation with missing data using powerfactorization and gpca," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. II-310–II-316.
[7] A. Mitiche, R. El-Feghali, and A.-R. Mansouri, "Motion tracking as spatio-temporal motion boundary detection," *Robot. Autonom. Syst.*, vol. 43, no. 1, pp. 39–50, 2003.
[8] R. Feghali and A. Mitiche, "Spatiotemporal motion boundary detection and motion boundary velocity estimation for tracking moving objects with a moving camera: a level sets pdes approach with concurrent camera motion compensation," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1473–1490, Nov. 2004.
[9] D. Cremers and S. Soatto, "Variational space-time motion segmentation," in *Proc. 9th IEEE Int. Conf. Computer Vision*, 2003, vol. 2, pp. 886–893.
[10] M. Ristivojevic and J. Konrad, "Space-time image sequence analysis: Object tunnels and occlusion volumes," *IEEE Trans. Image Process.*, vol. 15, pp. 364–376, 2006.
[11] L. Torres, M. Kunt, and F. Pereira, "Second generation video coding schemes and their role in mpeg-4," in *Proc. MPEG-4 Eur. Conf. Multimedia Applications, Services and Techniques*, 1996, pp. 799–824.
[12] M. Irani and P. Anandan, "About direct methods," in *Proc. Int. Workshop on Vision Algorithms: Theory and Practice*, 2000, pp. 267–277.
[13] R. Tron and R. Vidal, "A benchmark for the comparison of 3-d motion segmentation algorithms," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[14] H. Sekkati and A. Mitiche, "Concurrent 3-d motion segmentation and 3-D interpretation of temporal sequences of monocular images," *IEEE Trans. Image Process.*, vol. 15, no. 3, pp. 641–653, Mar. 2006.
[15] R. Lublinerman, O. Camps, and M. Sznaier, "Dynamics based robust motion segmentation," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 1176–1184.
[16] R. Vidal, S. Sastry, and Y. Ma, "Generalized principal component analysis (gpca)," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1945–1959, 2005.

[17] H. Sekkati and A. Mitiche, "Joint optical flow estimation, segmentation, and 3D interpretation with level sets," *Comput. Vis. Image Understand.*, pp. 89–100, Aug. 2006.
[18] A. Mitiche and H. Sekkati, "Optical flow 3D segmentation and interpretation: A variational method with active curve evolution and level sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1818–1829, Nov. 2006.
[19] K. Schindler, U. James, and H. Wang, "Perspective n-view multibody structure-and-motion through model selection," in *Proc. 9th Eur. Conf. Computer Vision*, 2006, pp. 606–619.
[20] T. Li, D. Singaraju, R. Vidal, and V. Kallem, "Projective factorization of multiple rigid-body motions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–6.
[21] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Proc. Int. Workshop on Vision Algorithms: Theory and Practice*, 2000, pp. 278–294.
[22] D. Cremers, T. Kohlberger, and C. Schnörr, A. Heyden, Ed. *et al.*, "Nonlinear shape statistics in mumford-shah based segmentation," in *Proc. Eur. Conf. Computer Vision*, Copenhagen, Denmark, May 2002, vol. 2351, pp. 93–108.
[23] D. Cremers, S. J. Osher, and S. Soatto, "Kernel density estimation and intrinsic alignment for shape priors in level set segmentation," *Int. J. Comput. Vis.*, vol. 69, no. 3, pp. 335–351, Sep. 2006.
[24] D. Cremers, "Dynamical statistical shape priors for level set based tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1262–1273, Aug. 2006.
[25] H. Sekkati and A. Mitiche, "Joint dense 3D interpretation and multiple motion segmentation of temporal image sequences: A variational framework with active curve evolution and level sets," *Image Process.*, vol. 1, pp. 553–556, 2004.
[26] N. Ichimura, "A robust and efficient motion segmentation based on orthogonal projection matrix of shape space," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, vol. 2, pp. 446–452.
[27] Y. Li, D. Hatzinakos, and A. Venetsanopoulos, "A multi-frame, region-feature based technique for motion segmentation," in *Proc. Int. Conf. Image Processing*, 1999, vol. 1, pp. 11–15.
[28] S. Smith and J. Brady, "Asset-2: Real-time motion segmentation and shape tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 814–820, Aug. 1995.
[29] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conf.*, 1988, pp. 147–151.
[30] J. Shi, J. Shi, C. Tomasi, and C. Tomasi, "Good features to track," in *Proc. IEEE Computer Society Computer Vision and Pattern Recognition*, 1994, pp. 593–600.
[31] Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, pp. 91–110, Nov. 2004.
[32] M. Gelgon, P. Bouthemy, and J.-P. Le Cadre, "Recovery of the trajectories of multiple moving objects in an image sequence with a PMHT approach," *Int. J. Comput. Vis.*, vol. 23, no. 1, pp. 19–31, 2005.
[33] B. Horn, *Robot Vision (MIT Electrical Engineering and Computer Science)*. Cambridge, MA: The MIT Press, 1986.
[34] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," *Comput. Vis. Pattern Recognit.*, vol. 2, pp. 2195–2202, 2006.
[35] J. Wang, G. Eude, Q. Liu, and H. Lu, "Moving object segmentation using graph cuts," in *Proc. ICSP 7th Int. Conf. Signal Processing*, 2004, vol. 1, pp. 777–780.
[36] H. Xu, M. Kabuka, and A. Younis, "Automatic moving object extraction for content-based applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 796–812, Jun. 2004.
[37] P. Viswanath and K. Jayasurya, "A fast and efficient ensemble clustering method," in *Proc. 18th Int. Conf. Pattern Recognition*, 2006, vol. 2, pp. 720–723.
[38] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res*, vol. 3, pp. 583–617, 2003.
[39] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 835–850, 2005.
[40] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 603–619, 2002.
[41] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, 1999.
[42] T. Meier and K. Ngan, "Automatic segmentation of moving objects for video object plane generation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 525–538, 1998.
[43] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 11, pp. 1442–1449, 2002.

Luciano S. Silva received the B.Eng. degree in computer engineering from the Federal University Foundation of Rio Grande, Rio Grande, RS, Brazil, in 2003, and the M.Sc. degree in computer science from the Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, in 2005. Currently, he is pursuing the Ph.D. degree in computer science at the Federal University of Rio Grande do Sul.

He is a Lecturer at the Federal University Foundation of Rio Grande.



Jacob Scharcanski (SM'03) received the B.Eng. degree in electrical engineering in 1981 and the M.Sc. degree in computer science in 1984, both from the Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil, and the Ph.D. degree in systems design engineering from the University of Waterloo, Waterloo, ON, Canada, in 1993.

His main areas of interest are image processing, pattern recognition, and computer vision. He has authored and coauthored more than 90 papers in journals and conferences. He also held research and development positions in the Brazilian and North American Industry. Currently, he is an Associate Professor with the Institute of Informatics and also has a cross appointment at the Electrical Engineering Department, both at the Federal University of Rio Grande do Sul. He is also an Adjunct Professor with Systems Design Engineering Department, University of Waterloo.