Support Vector Motion Clustering

Isah A. Lawal, Fabio Poiesi, Davide Anguita and Andrea Cavallaro

***Abstract*—We present a closed-loop unsupervised clustering method for motion vectors extracted from highly dynamic video scenes. Motion vectors are assigned to non-convex homogeneous clusters characterizing direction, size and shape of regions with multiple independent activities. The proposed method is based on Support Vector Clustering (SVC). Cluster labels are propagated over time via incremental learning. The proposed method uses a kernel function that maps the input motion vectors into a high-dimensional space to produce non-convex clusters. We improve the mapping effectiveness by quantifying feature similarities via a blend of position and orientation affinities. We use the Quasi-conformal Kernel Transformation to boost the discrimination of outliers. The temporal propagation of the clusters' identities is achieved via incremental learning based on the concept of feature obsolescence to deal with appearing and disappearing features. Moreover, we design an on-line clustering performance prediction algorithm used as a feedback (closed-loop) that refines the cluster model at each frame in an unsupervised manner. We evaluate the proposed method on synthetic datasets and real-world crowded videos, and show that our solution outperforms state-of-the-art approaches.**

***Index Terms*—Unsupervised motion clustering, Quasiconformal Kernel Transformation, on-line clustering performance evaluation, crowd analysis.**

I. Introduction

Clustering motion vectors into groups with homogeneous spatial and directional properties [1]–[4] can promote anomaly detection in crowd activities [5] and identify the formation of congested areas [6]. The knowledge of crowd motion direction can also help multi-object tracking [7], walking route prediction [8] and pedestrian re-identification across cameras [9].

Crowd motion can be structured [6] or unstructured [4]. The former involves high-density crowds whose flow can be modeled as a liquid in a pipe and predicted by observing an instance of the video scene [6]. Structured crowds can be regarded as a convex problem[1] as the motion of individuals is constrained by the main crowd flow. Unstructured crowds are characterized by a lower density and interwoven motion vectors with individuals whose paths are more challenging to predict [4]. These tend to be highly non-convex as sub-groups of individuals can freely move in the space.

Clustering can be performed by fitting Gaussian Mixture Models to motion vectors directly [11] or by mapping motion vectors to tailored subspaces [3]. Clusters of coherently moving people can also be isolated in a crowd [4], [12] by temporally correlating tracklets, generated for example as Kanade-Lucas-Tomasi short tracks [13], so as to automatically highlight groups moving together. However, this approach may omit smaller groups with weak coherent motions. Crowd motion can also be characterized in terms of coherent or intersecting motion via Deep Networks [14]. Moreover, this method requires supervision and does not spatially localize activities. Deep Networks could also be used to infer clusters of features in an unsupervised manner as a dimensionality reduction problem (auto-encoders) [15].

In this paper we propose a solution to the non-convex crowd motion clustering problem in the input space. We first map features in a high-dimensional kernel space (feature space) and then solve the non-convex clustering problem. Because motion vectors may not be linearly separable or homogeneously distinguishable in the input space, we use Support Vector Clustering (SVC) [16] that can map feature vectors to a higher dimensional kernel space using a kernel function in order to facilitate separability [17]. SVC does not require initialization, or prior knowledge of the number and the shape of the clusters. The sparse representation of the clusters produced by SVC in the form of Support Vectors allows us to employ incremental learning in order to perform the temporal update of the cluster model. Because we are dealing with temporally evolving motion vectors (data streams), our framework accounts for the intrinsic temporal dependences of the data to improve clustering. Our method is therefore applicable to both structured and unstructured crowds. To avoid the cluster model drifting we consider the concept of feature obsolescence during the model update. Noisy and interwoven motion vectors are deleted via the use of the Quasiconformal Kernel Transformation [18] that we embed in SVC to boost the discrimination of outliers by moving samples close to each other in the high-dimensional feature space. Moreover, we introduce a novel closed-loop on-line performance evaluation measure to quantify the homogeneity of the feature vectors in the clusters at each frame. The resulting similarity score is used as a feedback to the clustering algorithm to iteratively refine the clusters in an unsupervised manner. We evaluate the proposed method on different real-world crowd videos and compare its performance with state-of-the-art clustering methods. Project webpage: http://www.eecs.qmul.ac.uk/~andrea/svmc.html.

The paper is organized as follows. In Sec. II we survey state-of-the-art clustering methods along with clustering performance evaluation measures. We describe how the motion

I.A. Lawal, F. Poiesi and A. Cavallaro are with the Centre for Intelligent Sensing, Queen Mary University of London, Mile End Road, UK, E1 4NS, emails: fabio.poiesi@qmul.ac.uk, a.cavallaro@qmul.ac.uk. I.A. Lawal and D. Anguita are with University of Genoa, Via Opera Pia 11A, I-16145 Genoa, Italy, e-mail: abdullahi.lawalIsah@unige.it, davide.anguita@unige.it.

This work was supported in part by the Erasmus Mundus Joint Doctorate in Interactive and Cognitive Environments (which is funded by the EACEA Agency of the European Commission under EMJD ICE FPA n 2010-0012) and by the Artemis JU and the UK Technology Strategy Board through COPCAMS Project under Grant 332913.

[1]A cluster is convex in the input space (i.e. Euclidean space) if, for every pair of feature vectors within a cluster, every feature vector on the straight line segment that joins them is also within the cluster boundary [10].

cluster boundaries are computed and the incremental learning model in Sec. III and Sec. IV, respectively. In Sec. V we discuss the difference among our method and state-of-the-art alternatives. In Sec. V-A we analyze the computational complexity and in Sec. VI we evaluate the clustering performance of the proposed method. In Sec. VII we draw the conclusions and future research directions.

II. State of the art

In this section we discuss state-of-the-art feature clustering methods and performance evaluation measures that quantify the accuracy of clustering results.

A. Feature clustering methods

Feature clustering can be achieved using Watershed Clustering (WC) [19], Hierarchical Agglomerative Clustering (HAC) [20], Graph-based Normalized Cut (N-CUT) [21], Affinity Propagation (K-AP) [22], K-Means (KM) [23], Mean-Shift (MS) [24] or Dirichlet Process Mixture Model (DPMM) [25].

Watershed Clustering (WC) uses a grid over the input features to calculate a density function based on the distance among the features [19]. Cells with high feature similarity are selected as clusters. WC is sensitive to the selection of the cell size and its inaccurate selection could lead to an overestimation of the size and number of clusters.

Hierarchical Agglomerative Clustering (HAC) initially assumes each feature to be a single cluster and then iteratively merges clusters pairs based on their similarity [20]. HAC requires a user to specify the stopping criterion and the intra-cluster similarity for cluster merging. Moreover, HAC cannot generate motion clusters of arbitrary shapes that are key for the problem of motion clustering [26].

Graph-based Normalized Cut (N-CUT) models the set of input features as a graph, where nodes represent features and edges represent the similarity between features [21]. The graph is iteratively partitioned into clusters. Similarly to HAC, N-CUT requires a stopping criterion to be provided at initialization, thus reducing its applicability in cases of feature streams (e.g. feature vector clustering of videos).

Affinity Propagation (K-AP) computes a pair-wise similarity between input features and generates the clusters as the K most similar groups. As for N-CUT and HAC, the number of expected clusters needs to be defined *a priori*.

K-Means (KM) randomly initializes K input features as cluster centroids and then computes the distance among features and centroids [27]. Features are assigned to the clusters with the closest centroid. Once all features are assigned to clusters, the centroids of the clusters are recalculated. This procedure is repeated until all cluster centroids stop varying. KM requires the number of clusters to be specified in advance and assumes that clusters are convex in the input space.

Mean-Shift (MS) can automatically determine the number of motion clusters and does not use any constraints on their shape. MS performs clustering by computing the mean of the features lying within the same kernel. The kernel is iteratively shifted to the computed mean and the process is repeated until convergence. The selection of the kernel size is needed, and its inaccurate selection could lead to over- or under-estimating the number of clusters.

TABLE I
Comparison of the main properties of motion vector clustering methods.

Properties	WC [19]	HAC [20]	K-M [23]	N-CUT [21]	K-AP [22]	MS [24]	CF [4]	SVC [16]	**Ours**
Unsupervised	✓							✓	✓
Online performance evaluation									✓
Unknown number of clusters at initialization	✓	✓		✓	✓	✓	✓	✓	✓
Robustness to outliers				✓		✓	✓		✓
Can produce arbitrarily shaped clusters	✓					✓	✓	✓	✓
Iterative cluster refinement		✓	✓	✓		✓	✓		✓

Dirichlet Process Mixture Model (DPMM) assumes that the input features are generated from a Mixture of Gaussians and clusters are represented by the set of parameters of the mixture [25]. DPMM is limited by the assumption of Gaussian clusters' kernels, thus not allowing non-convex shapes nor input features with elements lying in different spaces.

SVC [16] performs clustering by using a non-linear kernel function to map feature vectors from input space into a high-dimensional feature space and then constructs a hypersphere enclosing the mapped feature vectors. This hypersphere, when mapped back to the input space, separates into several cluster boundaries defined by those data points known as the Support Vectors (SVs), each enclosing a different cluster. SVC does not require to specify the number of clusters in advance as it can infer this number during the optimization. However, an appropriate kernel and regularization parameters need to be selected for clustering. SVC builds the cluster model using batches of feature vectors and does not have an update mechanism that allows the temporal adaptation of such a model.

Table II-A compares the properties of state-of-the-art clustering methods with those of our proposed approach.

B. Performance evaluation measures

The performance evaluation of clustering results is commonly carried out using Normalized Mutual Information [28], Rand Index [29] or Dunn Index [30].

The Normalized Mutual Information (NMI) quantifies the extent of predicted cluster labels (PCL) with respect to the desired cluster labels (DCL) (i.e. ground truth). NMI is computed as the mutual information between PCL and DCL, normalized by the sum of the respective entropies.

The Rand Index (RI) measures how accurately a cluster is generated by computing the percentage of correctly labeled features. The accuracy score is given by the sum of true positive and true negative PCLs (normalized by the sum of all true and false PCLs) averaged over the whole PCL set. True positives are the number of correct PCLs with respect to a ground truth, whereas true negatives are the number of features correctly classified as outliers. The value of NMI and RI scores lies in the interval $[0, 1]$. The larger the NMI or RI score, the better the clustering. However, to quantify clustering performance on-line it is not possible to use ground-truth based evaluation measures such as NMI or RI.

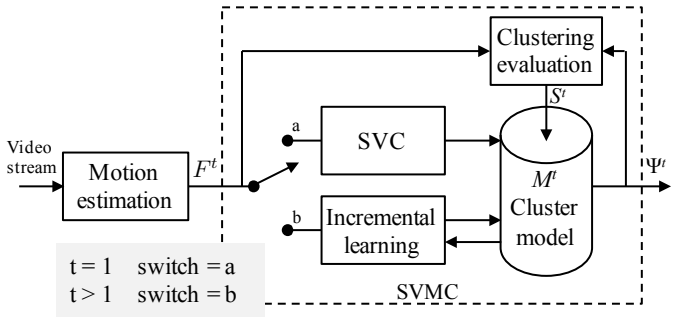


Fig. 1. Block diagram of the proposed approach (SMVC - Support Vector Motion Clustering). We first estimate the cluster boundaries and then use a Quasiconformal Kernel Transformation along with the on-line performance evaluation to refine the cluster boundaries. Then, we label each cluster and assign these labels to the respective feature vectors. M^t defines the cluster model at a generic time t, Ψ^t is the set of clusters and $\mathcal{S}^t$ is the Dissimilarity score that quantifies the clustering performance.

The Dunn Index (DI) quantifies clustering performance without ground-truth information. DI evaluates whether features are compactly included in clusters and clusters are clearly separated by computing the ratio between the maximum (Euclidean) distance between features belonging to the same clusters (cluster compactness) and the minimum (Euclidean) distance between features belonging to different clusters (cluster separation). The value of DI is not defined within a fixed interval. The lower the DI value, the better the clustering performance. However, we will need a performance measure that is unconstrained by the shapes of the clusters in order to quantify clustering performance on-line. For this reason, we will design a new evaluation measure, which does not need ground-truth information and can cope with sparse features and clusters with irregular shapes.

III. Motion cluster boundaries

A. Overview

Let $F^t = \{\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{N^t}^t\}$ be a set of motion vectors (feature vectors) extracted from a video stream at frame t, with $\boldsymbol{x}_i^t \in \mathbb{R}^d$, $i = 1, \ldots, N^t$, where $\boldsymbol{x}_i^t = [x^t, y^t, \dot{x}^t, \dot{y}^t]$ is characterized by its 2D position $[x^t, y^t]$ and velocity $[\dot{x}^t, \dot{y}^t]$. N^t is the cardinality of F^t and $d = 4$ is the dimension of $\boldsymbol{x}_i^t$.

Our objective is to learn a cluster model M^t that defines a set of distinctive clusters of F^t at each t *without prior information on the shape and number of clusters*.

We use Support Vector Clustering (SVC) to generate the initial ($t = 1$) set of clusters Ψ^1 of F^1 belonging to M^1. We then temporally update M^t for $t > 1$ via incremental learning by taking into account the *obsolescence* of features: old features are deleted and newer ones are used to update M^t. We use a Dissimilarity score $\mathcal{S}^t$ to evaluate the clustering performance and refine the model M^t for each t. Fig. 1 shows the block diagram of our proposed approach.

B. Support Vectors

Feature vectors are generally distributed in a way that their intrinsic similarities cannot be easily quantified in the input space as they are non-linearly separable [31]. For this reason, we use a non-linear transformation $\phi(\cdot)$ to map each feature $\boldsymbol{x}_i^t \in F^t$ to a high-dimensional non-linear feature space where the feature vectors and Support Vectors are separable. The mapping $\phi(\boldsymbol{x}_i^t)$ need not to be computed explicitly as we can implicitly quantify the feature similarity in the feature space (via the so-called *kernel trick*). Such a feature similarity allows us to determine the Support Vectors by generating a hypersphere with center c^t and radius R^t that encloses feature vectors of the same clusters in the feature space. The Support Vectors are features that define the boundaries of each cluster and can then be mapped back to the input space to form the set of clusters' boundaries

The hypersphere is determined via the following minimization problem [32]:

$$\begin{aligned} \mathbb{M} &= \underset{R^t, \xi_i^t, c^t}{\arg\min} \left\{ (R^t)^2 + \sum_{i=1}^{N^t} W_i^t \xi_i^t \right\} \\ \text{subject to} \quad & \left\| \phi(\boldsymbol{x}_i^t) - c^t \right\|^2 \leq (R^t)^2 + \xi_i^t, \quad i = 1, \ldots, N^t, \end{aligned} \tag{1}$$

where $\mathbb{M}$ contains the arguments R^t, ξ_i^t and c^t that minimize the function and $\phi : \mathbb{R}^4 \rightarrow \mathcal{H}$ is a non-linear function that maps $\boldsymbol{x}_i^t \in \mathbb{R}^4$ to the feature space $\mathcal{H}$, where the dimensionality of $\mathcal{H}$ is infinite for the case of a Gaussian kernel. ξ_i^t are slack variables that allow spurious $\boldsymbol{x}_i^t$ (outliers) to lie outside the hypersphere and W_i^t are weights that regulate the number of outliers [32].

Because $\phi(\boldsymbol{x}_i^t)$ has infinite dimension, Eq. 1 cannot be solved directly. A solution is to transform Eq. 1 into its Wolfe dual form [33]:

$$\begin{aligned} \mathbb{W} &= \underset{\beta_i}{\arg\max} \quad \sum_{i=1}^{N^t} \beta_i K(\boldsymbol{x}_i^t, \boldsymbol{x}_i^t) - \sum_{i=1}^{N^t} \sum_{j=1}^{N^t} \beta_i \beta_j K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) \\ \text{subject to} \quad & \sum_{i=1}^{N^t} \beta_i = 1, \quad 0 \leq \beta_i \leq W_i, \quad i = 1, \ldots, N^t, \end{aligned} \tag{2}$$

where $\mathbb{W}$ is the set of $\beta_i \forall i = 1, \ldots, N^t$ that satisfy the maximization, which can be solved via Sequential Minimization Optimization (SMO) [34].

$K(\cdot)$ is the kernel function that computes the inner product between $\phi(\boldsymbol{x}_i)$ and $\phi(\boldsymbol{x}_j)$ as

$$K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) = \left\langle \phi(\boldsymbol{x}_i^t) \cdot \phi(\boldsymbol{x}_j^t) \right\rangle = e^{-q^t \cdot d(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)}, \tag{3}$$

where $d(\cdot)$ is a function that computes the similarity between components (pair-wise similarity) of the feature vectors and q^t is a time-dependent coarseness parameter. q^t controls the extent of the cluster boundaries (cluster size) and (implicitly) the number of generated clusters [16]. This parameter is usually manually chosen and the automatic selection of q^t is still an open question [35]–[38].

C. Pair-wise similarity

In Eq. 3 $d(\cdot)$ measures the pair-wise similarity among feature vector elements. In the traditional SVC, $d(\cdot)$ is the Euclidean distance $\mathcal{D}_e(\cdot)$ [16], but this is not suitable to

measure angular distances as in our case. For this reason, we include the Cosine distance into the SVC kernel function to measure the orientation similarity [39] and we propose to use it along with the Euclidean distance, which is applied to the spatial coordinates.

The Euclidean distance is normalized as

$$\mathcal{D}_e([x_i^t, y_i^t], [x_j^t, y_j^t]) = \frac{1}{\Delta} \left\| [x_i^t, y_i^t] - [x_j^t, y_j^t] \right\|^2, \quad (4)$$

where Δ is the diagonal length of the video frame.

The Cosine distance is computed as

$$\mathcal{D}_c([\dot{x}_i^t, \dot{y}_i^t], [\dot{x}_j^t, \dot{y}_j^t]) = \frac{1}{2} \left(1 - \frac{[\dot{x}_i^t, \dot{y}_i^t] \cdot [\dot{x}_j^t, \dot{y}_j^t]}{\|[\dot{x}_i^t, \dot{y}_i^t]\| \, \|[\dot{x}_j^t, \dot{y}_j^t]\|} \right). \quad (5)$$

We combine these two normalized measures as

$$d(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) = \frac{1}{2} \left(\mathcal{D}_e \left([x_i^t, y_i^t], [x_j^t, y_j^t] \right) + \mathcal{D}_c([\dot{x}_i^t, \dot{y}_i^t], [\dot{x}_j^t, \dot{y}_j^t]) \right), \quad (6)$$

where $d(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) \in [0, 1]$.

D. Closed-loop update of the coarseness parameter

To enable unsupervised clustering, we design a method to automatically estimate the value of q^t at each t. q^t is used to determine the structure of the hypersphere that contains the feature vectors mapped in the feature space. This is achieved with a closed-loop solution that quantifies the similarity of feature vectors belonging to a cluster. Such a similarity is fed back to the clustering algorithm to determine (online) the performance for a certain q^t.

Authors in [40], [41] show that the relationship between q^t and the number of clusters induced when the hypersphere is mapped back to the input space is a piecewise constant function [42]. That is, for any arbitrary feature vector set, there exist some interval of q^t for which the values of the hypersphere are stable [41], thus leading SVC to produce the same number of clusters but with a slightly different shape boundary and size. Based on this notion we automatically select a suitable q^t from a given set by exploiting these piecewise constant intervals to choose the value of q^t via a measure that estimates the homogeneity of a cluster.

We first compute the set of possible values of q^t as

$$\left[q_{min}^t, q_{max}^t \right] = \left[\frac{1}{\max_{i,j} d(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)}, \frac{1}{\min_{i,j,i \neq j} d(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)} \right], \quad (7)$$

where q_{min}^t produces a single large cluster enclosing all feature vectors [16] and $q^t = q_{max}^t$ leads to a number of clusters equal to the number of feature vectors. Figure 2 shows an example where the clustering is performed with $q^t = q_{min}^t$ and with larger values of q^t, which increase the number of clusters.

The best value of q^t can be determined by iteratively analyzing the clustering performance. In particular, because feature vectors are unlabeled, we design a measure to perform an unsupervised self-assessment of the clustering performance namely Dissimilarity Score $\mathcal{S}^t$ that quantifies the homogeneity of the feature vectors belonging to each specific cluster.

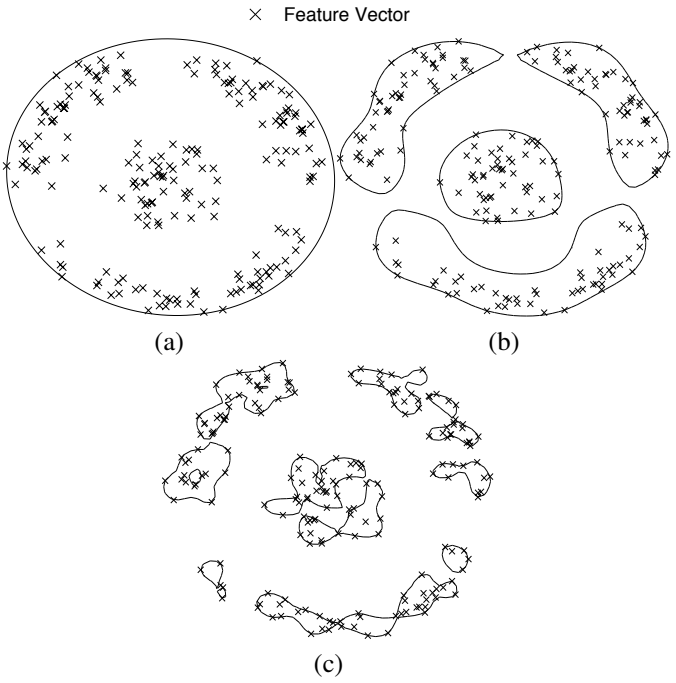


Fig. 2. Example showing three different clustering steps with (a) $q = q_{min} = 0.2$, (b) $q = 10$ and (c) $q = 25$. Only one cluster is generated when $q = q_{min}$ and the number of clusters increases when the value of q increases.

E. Dissimilarity score

$\mathcal{S}^t$ is computed as similarity of feature vectors at neighborhood level and then extended to cluster level. Specifically, the inner orientation coherency s_n^t of each cluster $h_n^t \in \Psi^t$ (the set of all clusters) is quantified using the circular variance of local feature vector orientations [39].

The Dissimilarity Score $\mathcal{S}^t$ is then calculated as the weighted average of s_n^t among all the clusters:

$$\mathcal{S}^t = \frac{\sum_{n=1}^{|\Psi^t|} |h_n^t| s_n^t}{\sum_{n=1}^{|\Psi^t|} |h_n^t|}. \quad (8)$$

The feedback loop involves an initial generation of meaningful candidate values for $q^t \in [q_{min}^t, q_{max}^t]$ achieved via Kernel Width Sequence Generator (KWSG) [37]. Let q_m^t, with $m = 1, ..., Q^t$, be the set of candidate values for q^t. Then, for each q_m^t, SVC generates the candidate clusters Ψ_m^t along with their $\mathcal{S}_m^t$. The final value of q^t is automatically selected within the interval where the computed number of clusters remains relatively stable (Fig. 3 - red ellipse) and the Dissimilarity Score $\mathcal{S}_m^t$ has minimum value within this interval. Fig. 4 shows the block diagram for the automatic generation of the coarseness parameter.

F. Refined Support Vectors

The solution of Eq. 2 provides the values of the Lagrangian multipliers β_i associated to each $\boldsymbol{x}_i^t$, which are used to classify feature vectors as either Support Vectors or Bounded Support Vectors. Specifically, the $\boldsymbol{x}_i^t$ for which $0 < \beta_i < W_i^t$ is a *Support Vector* S_v^t; $\phi(\boldsymbol{x}_i^t)$ lies on the surface of a hypersphere and describes the clusters' boundaries in the input space. The $\boldsymbol{x}_i^t$ for which $\beta_i = W_i^t$, is a *Bounded Support Vector* (B_v^t);

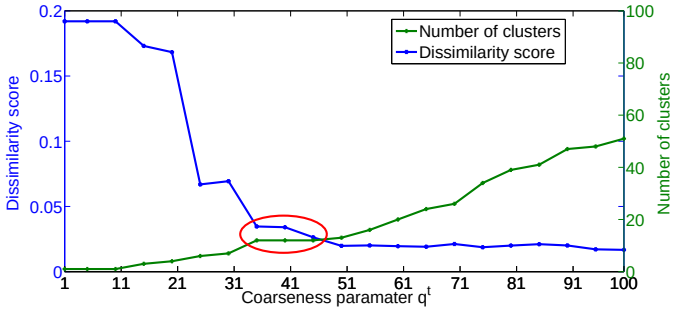


Fig. 3. Dissimilarity Score and number of clusters as a function of q^t. The red ellipse shows the range of q^t values in the case of a constant number of clusters. Within this interval we select the value of q^t corresponding to the smallest Dissimilarity Score.

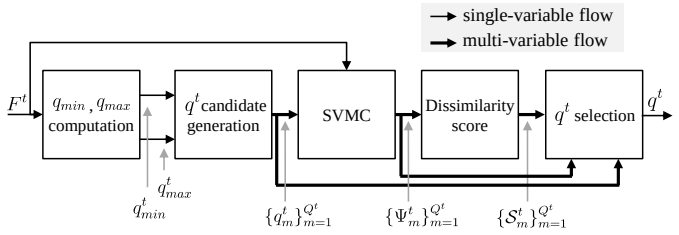


Fig. 4. Iterative process for the automatic generation of the coarseness parameter q^t using the input features F^t. SVMC: Support Vector Motion Clustering.

it is considered as an outlier because $\phi(\boldsymbol{x}_i^t)$ lies outside the surface of the hypersphere.

Feature vectors that lie on the clusters' boundaries may have been either assigned to a wrong cluster or wrongly classified as inliers. Note that the similarity among feature vectors in the high-dimensional space (feature space) is quantified via the kernel function $K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ (Eq. 3). To obtain a more accurate set of β_i values (Eq. 2) we employ the Quasiconformal Kernel Transformation [18] to modify $K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$. This transformation is typically used in a Support Vector Machine [18] and Nearest Neighbor Classification [43] to make local neighborhoods in a high-dimensional space more compact. The Quasiconformal Kernel Transformation promotes the deletion of outliers as it creates a new kernel $\tilde{K}(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ from $K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ by means of a kernel manipulation involving the Quasiconformal function $Q(\cdot)$. $\tilde{K}(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ can produce a hypersphere with an increased resolution that favors a higher discrimination between Support Vectors and Bounded Support Vectors (outliers). The construction of $Q(\cdot)$ is chosen such that the similarity between each pair of features (i.e. $\phi(\boldsymbol{x}_i^t)$ and $\phi(\boldsymbol{x}_j^t)$) is weighted by an exponential function that gives a greater penalization to features that are distant from each other.

We define a positive real-value function $Q(\boldsymbol{x}_i^t)$ for each $\boldsymbol{x}_i^t \in F^t$ [43]. We use $Q(\boldsymbol{x}_i^t)$ to scale $K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$ to $\tilde{K}(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)$, which in turn is used to solve Eq. 2 to obtain a new set of β_i and to re-estimate the Support Vectors.

$\tilde{K}(\cdot)$ is defined as

$$\tilde{K}(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) = Q(\boldsymbol{x}_i^t) K(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) Q(\boldsymbol{x}_j^t), \tag{9}$$

with

$$Q(\boldsymbol{x}_i^t) = \sum_{k=1}^{|S_v^t|} e^{-\left\|\phi(\boldsymbol{x}_i^t) - \phi(\boldsymbol{x}_k^t)\right\|^2 / \tau_k^2}, \quad \forall \boldsymbol{x}_k^t \in S_v^t, \tag{10}$$

where the free parameter τ_k is calculated as

$$\tau_k = \left(\frac{1}{z} \sum_{l=1}^{z} \left\|\phi(\boldsymbol{x}_l^t) - \phi(\boldsymbol{x}_k^t)\right\|^2 \right)^{\frac{1}{2}}, \forall l, k \in S_v^t,\ l \neq k, \tag{11}$$

z is the number of nearest Support Vectors to $\boldsymbol{x}_k^t$ in S_v^t.

The new set of Support Vectors that defines the clusters' boundaries is

$$\mathcal{C}_c^t = \left\{ \boldsymbol{x}_i^t | \mathcal{R}(\boldsymbol{x}_i^t) = \max_{\boldsymbol{x}_i^t \in S_v^t} (\{\mathcal{R}(\boldsymbol{x}_i^t)\}) \right\}, \tag{12}$$

where $\mathcal{R}(\cdot)$ is the distance of each feature vector from the center of the hypersphere defined as

$$\mathcal{R}(\boldsymbol{x}_k^t) = \left(1 - 2 \sum_{i=1}^{N^t} \beta_i \tilde{K}(\boldsymbol{x}_i^t, \boldsymbol{x}_k^t) + \sum_{i=1}^{N^t} \sum_{j=1}^{N^t} \beta_i \beta_j \tilde{K}(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) \right)^{\frac{1}{2}}. \tag{13}$$

The generation of new Support Vectors (defining the clusters' boundaries) allows us to define the cluster affiliation of each feature vector. Therefore, we label the clusters and assign each $\boldsymbol{x}_i^t \in F^t$ to its affiliated cluster using the Complete Graph (CG) labeling method [16]. In CG a pair of feature vectors $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^t$ is said to belong to the same cluster (connected components) if the *line segment* (inside the feature space) that connects them lies inside the hypersphere. A number of feature vectors (usually 10 [16]) are sampled on the *line segment* to assess the connectivity. We use 10 sampled feature vectors as this number provides cluster labelling results with a negligible error along with a limited computational complexity [16]. A higher accuracy can be achieved with a larger number of sampled feature vectors, but with the disadvantage of a multiplicative increase in the computation time [44].

Therefore, we construct an adjacency matrix $A^t = [A_{i,j}^t]^{N^t \times N^t}$ where $A_{i,j}^t = 1$ if $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^t$ are connected components. Each $A_{i,j}^t$ is defined as

$$A_{i,j}^t = \begin{cases} 1 & \text{if } \mathcal{R}(\boldsymbol{x}_i^t + \lambda(\boldsymbol{x}_i^t - \boldsymbol{x}_j^t)) \leq R,\ \forall \lambda \in [0,1] \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where λ is a free parameter used to account for the number of sampled feature vectors along a line segment. As in [16], we use λ = 0.1 to allow 10 feature vectors to be sampled. As A^t is a graph of connected components, we label each independent sub-graph and assign the resulting labels to the appropriate feature vectors to define their affiliation.

We denote each cluster at t as h_i^t, the set of all clusters as Ψ^t and the set of indexes of the clusters in Ψ^t as Ω^t. The cluster model M^t produced at t is then defined as

$$M^t = \left(\Psi^t, \Omega^t\right). \tag{15}$$

In the next section we discuss how this model is updated over time.

IV. Cluster model update

A new set of feature vectors F^t is to be clustered for each $t > 1$. We update M^{t-1} to M^t by using Support Vectors and their cluster affiliation in previous frames. For each F^t we generate the Support Vectors, the Bounded Support Vectors and the new feature vector set $\hat{F}^t = S_v^t \cup B_v^t \cup F^t$.

Traditional incremental learning approaches ([25], [45], [46]) address the problem of the cluster model update containing features "sampled" from the same source with an invariant generative distribution. This means that at initialization the clusters are not fully defined and the incremental learning step updates the cluster model based on the new spawned features (sampled from the same distribution).

However, in our case the distribution of features in the input space is time variant: features vary, appear and disappear anywhere in the state space because of the changing dynamics of people moving. Thus some Support Vectors and Bounded Support Vectors of previous frames may be obsolete and, if used to update M^{t-1}, they would lead to inaccurate clusters.

To address this problem, we monitor the life span of each Support Vector, and update B_v^t and S_v^t as functions of previous frames. Specifically, Bounded Support Vectors may become part of (or leave) some clusters or generate new clusters themselves due to spawning or disappearing motion vectors. The Bounded Support Vectors within the temporal interval $[t - T_{min}, t]$ that failed to become Support Vectors or inliers (after T_{min} frames) are considered persistent outliers and eliminated. T_{min} defines the deadline for which Bounded Support Vectors are considered for in the incremental step. The larger T_{min}, the longer the temporal window in the past outliers are considered in the incremental step.

Support Vectors may also change because the shape of the clusters changes due to spawning or disappearing objects in the crowd, or due to changes in the crowd structure. If the Support Vectors are not updated or eliminated according to the evolution of the motion vectors, they would lead to under-estimated clusters. Therefore, we eliminate Support Vectors that are unused for more than T_{max} frames (within $[t-T_{max}, t]$). Similarly to T_{min}, T_{max} defines the deadline for which the Support Vectors are considered in the incremental step. The larger T_{max}, the larger the clusters' boundaries that will encompass the motion vectors. In videos it can be equal to the value of the frame rate (e.g. 25 frames).

The feature vectors within $\hat{F}^t$ are then used to solve Eq. 2, which provides the updated set of Support Vectors that defines the clusters' boundaries C_c^{t+1} (Eq. 12) and the updated set of Bounded Support Vectors that defines the outliers.

Cluster labels are then assigned to all $\boldsymbol{x}_i^t \in F^t$ by recomputing the adjacency matrix A^t and re-labeling the graphs of the connected component (Eq. 14). Because during the incremental learning of M^{t-1} some old cluster boundaries might have become enlarged, due to the inclusion of new feature vectors $\boldsymbol{x}_i^t$, we propagate the same old clusters labels to the new $\boldsymbol{x}_i^t$.

We introduce a temporal cluster affiliation check to examine the connectivity of the feature vectors within a given cluster based on the adjacency matrix A^t in order to propagate the same cluster labels. The four cases are:

(i) *Unchanged cluster*. We check whether the graphs of the connected components induced by A^t are formed by the feature vectors from old clusters. This operation is performed by checking the ages of all the feature vectors within the graphs and their corresponding cluster indexes assigned to the previous frame. Then we maintain the same old cluster label for the feature vectors (i.e. no new clusters are formed).

(ii) *Enlarged cluster*. We check whether some graphs of the connected components of A^t are formed by the feature vectors from an old cluster, and the rest of the graphs are formed by the feature vectors in F^t. We propagate the same older cluster label to the newly included feature vectors.

(iii) *New cluster*. We check whether all the graphs of the connected components of A^t are formed by the feature vectors in F^t only, and then assign a new cluster label to the feature vectors.

(iv) *Merged cluster*. We check whether the connected components of A^t are formed by feature vectors from different old clusters, and then we assign to all the feature vectors the label of the old cluster with the largest number of feature vectors before the merging.

This four-case cluster assignment is summarized in Algorithm 1.

Algorithm 1 *Cluster assignment*

t: frame
Ψ^t: set of clusters
h_c^t: c^{th} cluster in Ψ^t
Ω^t: set of clusters' indexes
ω_c^t: index of cluster h_c^t
A^t: adjacency matrix
a_c^t: c^{th} graph of connected components in A^t

```
for all a_c^t do
  for all x_i^t do
    % Unchanged cluster & Enlarged cluster
    if x_i^t ∈ a_c^t ∧ (x_i^{t-1} ∈ h_c^{t-1} ∨ ∄x_i^{t-1}) then
      x_i^t ← ω_c^{t-1}
    end if
    % New cluster
    if x_i^t ∈ a_c^t ∧ ∄a_c^{t-1} then
      x_i^t ← ω_c^t
    end if
    % Merged cluster
    if x_i^t ∈ a_c^t ∧ x_i^{t-1} ∈ h_{c'}^{t-1} then
      h_c^t ← h_c^t ∪ h_{c'}^t
      x_i^t ← ω_c^t
    end if
  end for
end for
```

V. Discussion

A. Computational complexity

We analyze the computational complexity per frame and consider the two main components of the approach: the estimation of the clusters' boundaries over time and the cluster assignment of the feature vectors.

Let $\hat{F}^t = S_v^t \cup B_v^t \cup F^t$, be the set of feature vectors at frame t for $t > 1$, where S_v^t B_v^t and F^t are the Support Vectors, Bounded Support Vectors and newly obtained feature vectors at t, respectively. At each t, the update of M^{t-1} to M^t involves solving Eq. 2 using the SMO solver [34] with a complexity of $O((|\hat{F}^t|)^2)$ [47], followed by the cluster labeling procedure for the updated set of Support Vectors

describing the clusters' boundaries with a complexity of $O((|\hat{F}^t| - |B_v^t|)^2 |S_v^t|\lambda)$, where S_v^t and B_v^t are the updated Support Vector sets. λ is a free parameter. The computational complexity per frame is $O((|\hat{F}^t|)^2\lambda)\ \forall\ t$. For a video with a frame rate of 25fps, the computational burden is 25 times more.

We additionally analyze the per-frame computation time using three videos composed of 245, 500 and 900 frames, respectively. These videos contain 142 ± 20, 240 ± 38 and 175 ± 23 feature vectors on average per frame, respectively. SVMC is implemented in Matlab (the code is not optimized) and the experiments are run on a PC with Core Intel i5 2.50GHz and 6GB memory. The computation time per frame is $71.01 \pm 27.32s$, $142.23 \pm 57.80s$ and $89.09 \pm 35.66s$, respectively. The computed time comprises the time spent to update the cluster model M^t and the time used to perform the cluster labelling over the entire feature set. The cluster labelling accounts for the largest portion of the SVMC computation time, which includes generating the adjacency matrix and finding the connected components. The computation time of the cluster labelling can be reduced by employing algorithms such as SEP-Complete Graph (SEP-CG) [48], [49]. SEP-CG performs the cluster labelling by first identifying a set of stable equilibrium points (SEPs) that describe the cluster boundaries, which consist of feature vectors located around the same local minimum of the function (i.e. Eq. 14). Then the SEPs are employed to infer the connected components [48] via the use of additional feature vectors located on saddle-points [49]. The computational complexity of SEP-CG is $O(|\hat{F}^t| log |\hat{F}^t|)$ [48].

B. Comparison with respect to state of the art

We name the proposed approach as SVMC (Support Vector Motion Clustering). The differences between SVC and SVMC are: (i) SVC does not perform an unsupervised selection of the kernel parameter. The kernel parameter controls the hypersphere size and regulates the number of clusters formed; (ii) SVC calculates the multi-dimensional feature vector similarity by applying the same distance measure to all the feature elements. This can lead to clustering errors in the case of heterogeneous feature vector elements, as different elements might have to be compared with different distance measures. One case is motion clustering, where position and orientation features should be compared with Euclidean and angular distance, respectively; (iii) SVC does not have a mechanism for the model update, thus it is not suitable for data-stream clustering that require temporal the model update. SVC is designed to re-build the cluster model from scratch every time a new batch of feature vectors is to be computed. However, temporal dependencies need to be considered as they promote clustering accuracy; (iv) the concept of Quasiconformal Kernel Transformation is used for the first time with SVC framework to remove additional outliers, before it was used on Support Vector Machine with supervised problems.

Compared to other state-of-the-art algorithms (Sec. II). Unlike K-M, the number of the clusters need not to be defined in SVMC as it is automatically determined. Unlike N-CUT, K-AP and MS, SVMC is data-driven and unsupervised. Unlike DPMM, SVMC employs a distance measure that can deal with heterogeneous feature elements. Finally, unlike CF and CT, which require a strict spatial and temporal coherency of the feature vectors, SVMC relaxes this assumption and can include smaller clusters that still present coherent motion.

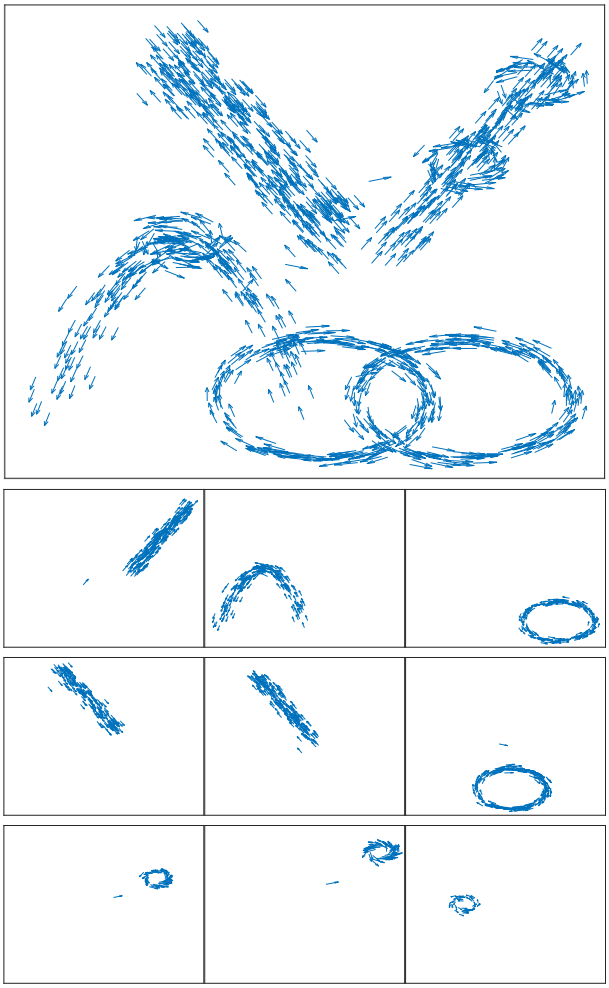


Fig. 5. A sample frame of the synthetic dataset with nine motion clusters used to evaluate the proposed method. The image resolution of the dataset is 467×366.

VI. Experimental Results

A. Datasets and evaluation

We evaluate the proposed clustering framework (Fig. 1) using a synthetic dataset and real-world crowd videos.

The synthetic dataset is generated with an extended version of the code used in [4] and includes nine different motion clusters for a total of 67500 feature vectors over 50 frames. The modification includes random variations of the velocity component, a larger number of feature vectors and additional motion clusters with different shapes (interwoven circular clusters). Each feature vector is characterized by position and velocity. We use this dataset in order to test the ability of our method in clustering homogenous motion flows with different scales, shapes and dynamics (Fig. 5 shows an instance of the synthetic dataset). We use 5 video sequences, namely *Marathon*, *Traffic*, *Train-station*, *Student003* and *Cross-walk*,

which include variations of crowd density. These are popular videos used for the evaluation of crowd motion segmentation in previous studies [3], [4], [6]. *Marathon* includes athletes running with coherent motion flow; the crowd density is high and structured. *Traffic* includes vehicles moving along two traffic lanes and some jaywalkers traversing these traffic lanes; the crowd is unstructured. *Train-Station* consists of pedestrians entering/exiting a train station; the crowd is unstructured. *Student003* involves a crowded student square where both individual and groups of pedestrians move in an unstructured manner. *Cross-walk* contains two large groups of pedestrians crossing each other on the crosswalk; the crowd is initially structured and then it changes to unstructured.

The clustering quality is evaluated using the proposed Dissimilarity Score $\mathcal{S}^t$ (Sec. III) and Normalized Mutual Information (NMI) [28], using the ground truth of the synthetic dataset for calculating NMI. We also compare the relative evaluation performance of $\mathcal{S}^t$ and NMI to understand whether the Dissimilarity Score agrees with the NMI assessment. We use the proposed Dissimilarity Score because our objective is also to compare it against NMI that is a popular state-of-the-art measure. NMI $\in$ [0,1], where $NMI \rightarrow 1$ indicates a better clustering performance. The Dissimilarity Score $\mathcal{S}^t \in [0, 1]$, where $\mathcal{S}^t \rightarrow 0$ indicates homogeneous feature vectors within clusters. The performance is evaluated over time to evaluate the incremental capabilities of the method.

We also quantitatively evaluate the clustering results using the method proposed in [2]: for each motion cluster in a video frame we count the number of individuals in the cluster whose motion direction does not differ more than 90 degrees from the mean motion direction of the individuals in the same motion cluster. The number obtained is considered as the number of correctly clustered people in the video frame. Thus, we compute the error rate of each method as the ratio of the number of incorrectly clustered people to the total number of people in a video frame [2].

B. Experimental setup and comparison

We use the Kanade-Lucas-Tomasi (KLT) tracker [13] to extract feature vectors from each frame; KLT feature vectors are characterized by a 2D position and velocity.

The choice of T_{min} and T_{max} is data and task dependent. Our choice of these two parameters is based on the following idea. T_{min} can be chosen according to the dynamics of the scene under consideration. We set $T_{min} = 2$ in order to allow feature vectors deemed to be Bounded Support Vectors at a certain time t to be re-considered as potential Support Vectors at the next time $t + 1$. A value of $T_{min} > 2$ would allow the incremental update to reconsider Bounded Support Vectors as Support Vectors for multiple time steps, but would lead to an increasing memory requirement and computational resources. T_{max} determines the time duration for which Support Vectors computed at a certain time step t are maintained over time to help the incremental update. We set $T_{max} = 25$ as we assumed that data do not largely change within 25 frames.

We compare our clustering results for the synthetic dataset with traditional SVC and six other state-of-the-art clustering methods: Graph-based Normalized Cut (N-CUT) [21], Affinity Propagation (K-AP) [22], Mean-Shift (MS) [24], Coherent Filtering (CF) [4], Streaming Affinity Propagation (StrAP) [50] and Incremental Dirichlet Process Mixture Model (IDPMM) [45]. We choose these for comparison as they are standard clustering methods [21], [22], [24], [45], [50] plus a recent specific method for motion clustering [4]. We set the kernel and regularization parameters of SVC to 2.5 and 1.0, respectively. We set the initial number of clusters to nine (equal to the number of clusters) for N-CUT and K-AP. StrAP, MS and IDPMM automatically determine the number of clusters. For CF we use the same parameters provided in [4].

We also compare motion clustering results against three other state-of-the-art methods for crowd motion clustering: Streak flows and watershed (Streakflow) [2], CF [4] and Collective Transition prior (CT) [12]. For this we use *Marathon*, *Traffic-junction*, *Train-station* and *Student003*. The same KLT feature vectors used in SVMC are given as input to CF and CT, while video frames are directly given as input to Streakflow as in the original implementation that uses post-processed optical flow is used instead of KLT feature vectors.

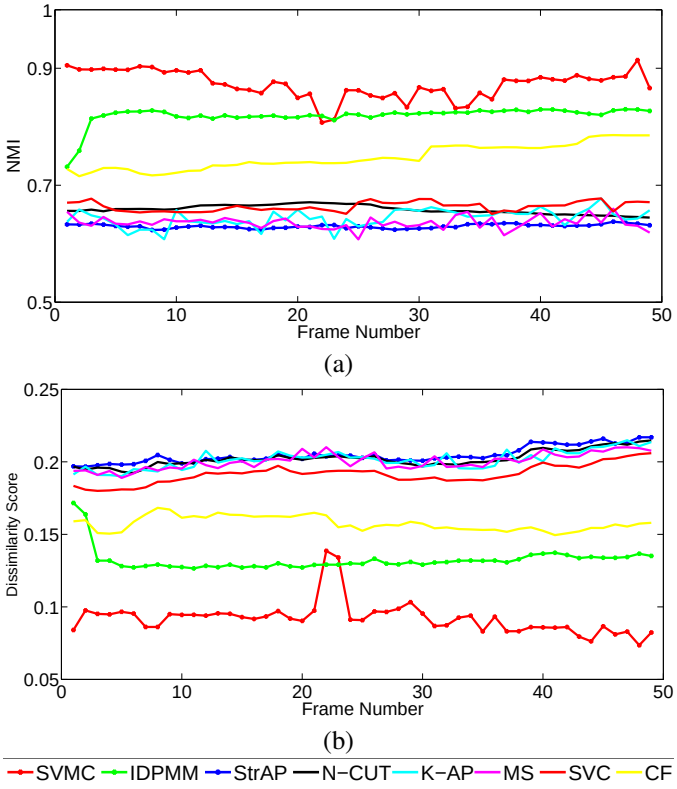


Fig. 6. NMI and Dissimilarity Scores for different clustering methods. Dotted lines: Incremental clustering methods. Continuous lines: Non-incremental clustering methods. (a) NMI $\in$ [0,1], $NMI \rightarrow 1$ indicates better clustering performance. SVMC (red dotted line) has the smallest average NMI score (0.8722) followed by IDPMM (green dotted line) with an average NMI score of 0.8190. (b) Dissimilarity Score $\in$ [0,1], $S^t \rightarrow 0$ indicates a better clustering performance. SVMC (red dotted line) has the smallest average Dissimilarity Score (0.0919), followed by IDPMM (green dotted line) with an average Dissimilarity Score of 0.1323.

C. Evaluation on the synthetic dataset

Fig. 6 shows NMI and $\mathcal{S}^t$ values for the compared state-of-the-art methods and the proposed approach. Fig. 7 shows a

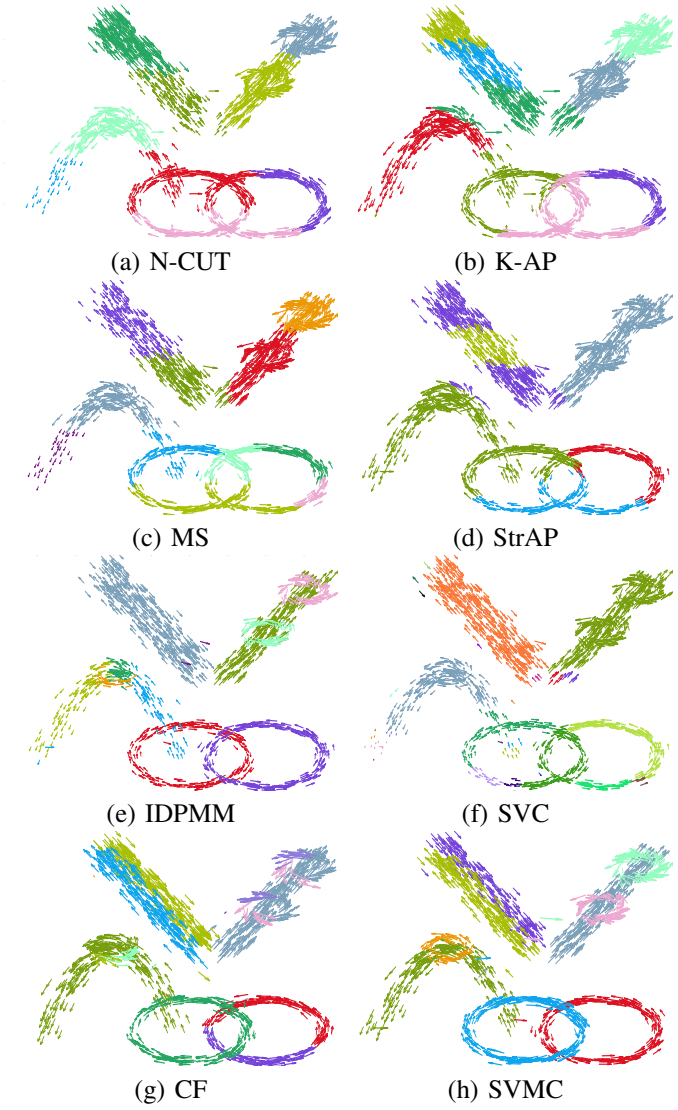


Fig. 7. Clustering results of the methods used for comparison on the synthetic dataset. The color-coded features represent the affiliation to the clusters. The SVMC is the only method that correctly estimates all the motion clusters. The resolution of each image is 467×366.

sample frame of these results. From Fig. 6 we can make two observations.

The first observation is that SVMC outperforms the other methods (i.e. highest NMI and lowest $\mathcal{S}^t$ overall) followed by IDPMM and CF. SVMC can effectively cluster features and maintain the best results over time. The transformation to a higher dimensional space allows SVMC to capture the intrinsic similarities among feature vectors of the same flow type although they appear interwoven to an observer (Fig. 7h). IDPMM is the second-best method as it can accurately cluster the circular patterns, but it is unable to separate the bi-directional flow on the top-left (Fig. 7e) because the algorithm cannot temporally propagate the similarity among features. Also, because the similarity among features is low, several feature vectors are deemed to be outliers. CF achieves the third-best results as it can accurately cluster flows with evident opposite directions (top-left Fig. 7g). Recall that CF produces clusters based on motion vector similarities in the input

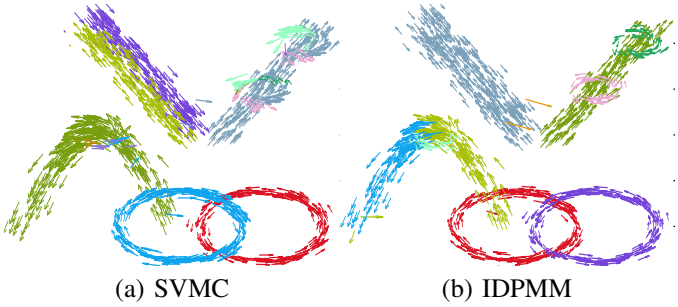


Fig. 8. Comparison of the results of (a) SVMC and (b) IDPMM at frame 22 of the synthetic dataset. Some feature vectors of the two small circular flows (top-right) are incorrectly assigned to different clusters by SVMC whereas IDPMM correctly clusters them. The resolution of each image is 467×366.

space with an iterative process that begins from a single motion vector. This design does not allow CF to determine overlapping motion vectors that could share similar patterns (motion vectors with the same direction localized in a certain area). In fact, CF fails in the case of small overlapping circular flows when the direction of some motion vectors coincides with that of the overlapped flow. SVC, N-CUT, K-AP and MS cannot effectively cluster the feature vectors as the direction component of the feature vectors is erroneously compared by the kernel of these methods. We can also observe that, unlike traditional SVC, N-CUT, K-AP and MS which do not have an online model updating mechanism, the incremental update of SVMC demonstrates its effectiveness in tracking the evolution of the clusters' boundaries over time and in propagating the same cluster identity to feature vectors belonging to the same flow. At frame 22 (Fig. 6) the NMI score and the Dissimilarity Score for SVMC show a drop in performance compared to IPDMM. This is because SVMC produces some errors when clustering the motion vectors of the circular flows during the incremental update (Fig. 8a, top-right circular flows).

In the specific case of SVMC improvements with respect to SVC, we can observe that SVMC can separate the two spatially close but opposite flows of the dataset in Fig. 7h (top-left flows - green and purple clusters) mainly because of the modified kernel that uses the hybrid distance measure that compares both the spatial and directional coordinate of the feature vectors. Whereas SVC clustered the two opposite flows as a single cluster (Fig. 7f top-left flow). Moreover, SVMC outperforms SVC as it can correctly cluster the feature vectors of the two circular flows into two clusters represented by the red and blue feature vectors (Fig. 7h bottom flows), because the incremental model update mechanism of SVMC can capture the temporal similarities of the coherent flows and can maintain the cluster identities associated to each feature vector over time. The same concept applies to the two circular flows interwoven with the straight flows on the top-right of Fig. 7h. SVC cannot separate them, whereas SVMC can.

We analyze the execution time of the compared methods on the synthetic dataset using the code provided by the authors. MS, CF, N-CUT, StrAP, IDPMM and K-AP achieve an average time of $0.08 \pm 0.03s$, $0.23 \pm 0.05s$, $0.28 \pm 0.14s$, $1.12 \pm 0.23s$, $6.54 \pm 0.77s$ and $139.00 \pm 32.01s$ per frame, respectively. SVC and SVMC are implemented by us with an un-optimized

Matlab code that achieves an average of $652.15 \pm 30.15s$ and $557.60 \pm 65.78s$ per frame. The incremental update allows us to save computation time as it does not need to re-compute the cluster model each time step.

The second observation is that NMI and $\mathcal{S}^t$ agree on the evaluation of the clustering performance. Note that $\mathcal{S}^t$ does not use annotated data for the analysis as opposed to NMI that does. For example, SVMC has the highest NMI (Fig. 6a) and the lowest $\mathcal{S}^t$ in Fig. 6b. Similarly, traditional SVC, N-CUT, K-AP, and MS which present low NMI, also have $\mathcal{S}^t$ comparably high. We can then infer that $\mathcal{S}^t$ can be used as a valid predictor for the clustering performance. From the graph we can also observe that the incremental learning does not drift over time and maintains the best performance throughout.

Finally, we further validate SVMC by using the original version of the synthetic dataset from CF [4] and compare our results with that obtained with the CF method. SVMC has NMI and $\mathcal{S}^t$ of 0.9820 and 0.0225, respectively, which are yet better than those of CF that are 0.9667 and 0.0248, respectively.

D. Comparison between SVMC vs. SVC on real crowd videos

Fig. 9 shows the Dissimilarity Score generated by SVMC and SVC, where SVMC outperforms SVC (Fig. 9a-c) with lower Dissimilarity Scores on all the videos. In *Train-station* and *Student003* the scenes are unstructured due to people moving incoherently and crossing each other, thus SVC without mechanism for evaluating the orientation of the different motions and for cluster refinement, cannot separate them in different clusters leading to a high Dissimilarity Score (red).

SVMC can instead cluster them with a lower Dissimilarity Score (green) because of the hybrid distance measure that discriminates motions with distinct directions and by means of the Quasiconformal Kernel Transformation that can generate more refined clusters than SVC. In the case of Cross-walk, there are two distinct sets of pedestrians crossing with opposite motion directions. The feature vectors generated from the scene until the 150^{th} frame can be easily clustered by SVC as the two sets of people are spatially far apart. However, when the two sets of people mix (i.e. from the 150^{th} frame onwards), the feature vectors become interwoven and thus difficult to assign to different clusters (Fig. 11a). The quality of the clusters generated by SVC begins to degrade as shown by the increasing Dissimilarity Score in Fig. 9c (red). The Dissimilarity Score for the clusters produced by SVMC remains instead low (thus indicating homogeneous clusters) throughout the video (Fig. 9c (green)).

We also compare the performance of SVMC and SVC using NMI and Dissimilarity Score on the same datasets using manually annotated clusters. We annotated 10 key-frames for each video, which contain sufficient variability of crowd structures. Fig. 10 shows the results in terms of average and standard deviation of the NMI and Dissimilarity Score on the 10 key-frames. Both NMI and Dissimilarity Scores show that SVMC provides better and more stable performance than the traditional SVC.

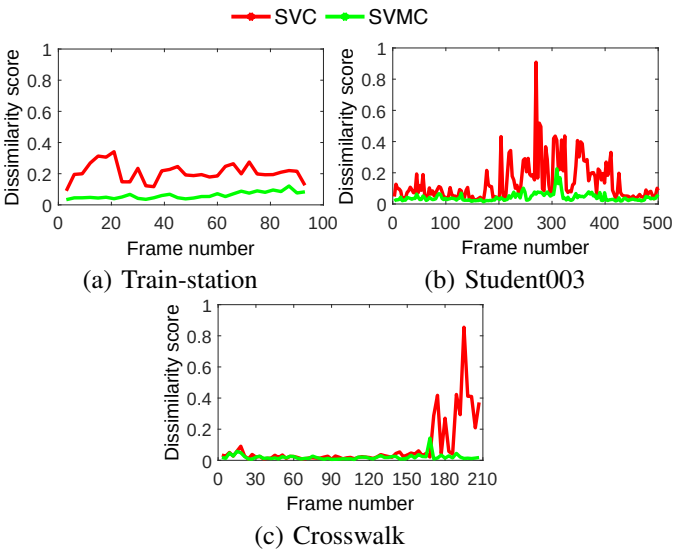


Fig. 9. Comparison of the Dissimilarity Scores generated by SVMC and SVC for (a) *Train-station*, (b) *Student003* and (c) *Cross-walk*. SVMC (green) produces clusters with feature vectors that are more coherent (i.e. lower Dissimilarity Score) than SVC (red).

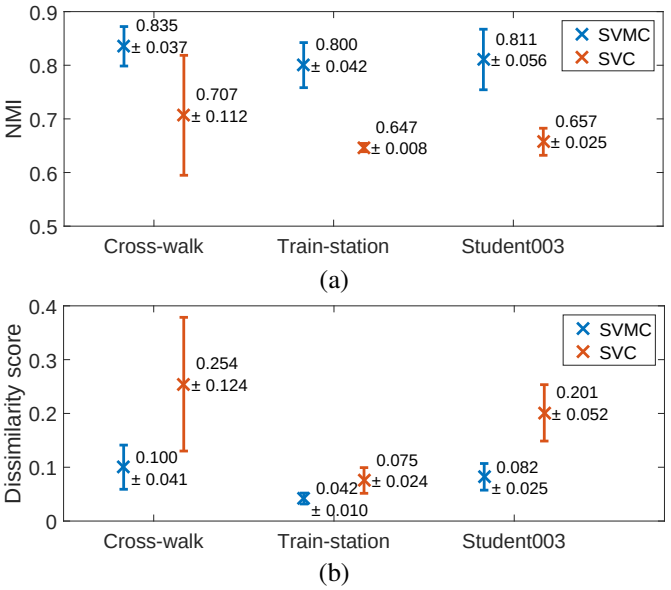
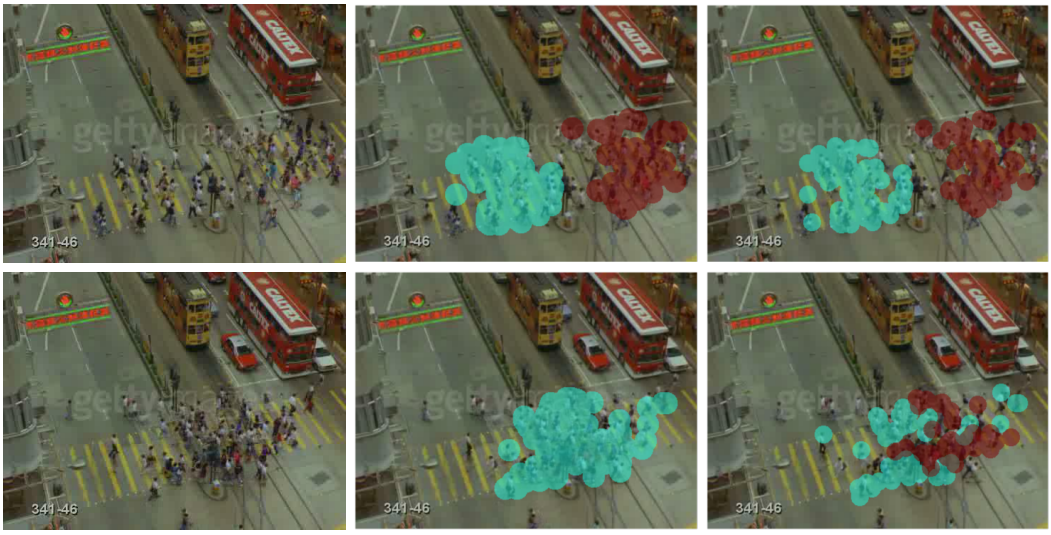


Fig. 10. Comparison between SVMC and SVC using (a) NMI and (b) Dissimilarity Score ($\mathcal{S}^t$) on Cross-walk, Train-station and Student003. The results are obtained as average and standard deviation of the NMI and Dissimilarity Score over 10 key-frames (manually selected and annotated). Note that $NMI \rightarrow 1$ and $\mathcal{S}^t \rightarrow 0$, signify better results. Results show that SVMC produces better and more stable performance than the traditional SVC, both with NMI and the Dissimilarity Score.

E. SVMC vs. other crowd motion clustering methods

Fig. 12 shows the quantitative comparison of the four motion clustering methods for two videos (*Student003* and *Train-station*), where the people in the scene are manually counted.

In *Student003*, SVMC outperforms Streakflow, CF and CT in terms of the number of correctly clustered pedestrians with an average error rate of 0.117 against the 0.164, 0.249 and 0.345 calculated for CF, Streakflow and CT, respectively. In *Train-station* the average error rates are 0.069 for SVMC,



(a) (b) (c)

Fig. 11. Sample clustering results on (a) a representative frame of *Cross-walk* for (b) SVC and (c) SVMC. The colored patches represent the motion clusters of people with the same motion direction. At the 50^{th} frame (top) both SVMC and SVC correctly cluster the two sets of people, at the 150^{th} frame (bottom) the SVC incorrectly clusters the two sets of people as one large cluster (cyan patch). SVMC can correctly cluster the two groups of intersecting people (i.e. cyan and dark red patches).

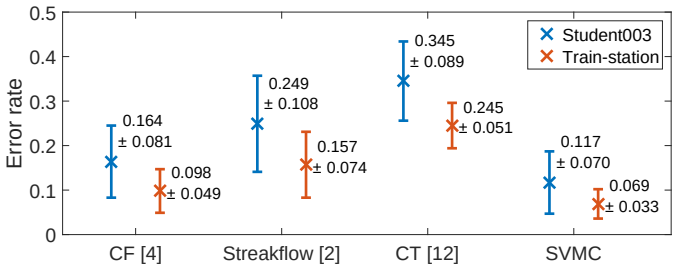


Fig. 12. Comparison of motion clustering error rates on *Student003* and *Train-station*. SVMC outperformed CF, Streakflow and CT in terms of number of correctly clustered people (i.e. lower error rate) for both videos.

0.098 for CF, 0.157 for Streakflow and 0.245 for CT. In general, the average error rate and standard deviation of SVMC are smaller than those of the other methods. This is mainly due to the precision introduced by the Quasiconformal Kernel Transformation with which SVMC can determine cluster boundaries and update them over time. Note that the update involves outlier deletion and computation of new Support Vectors.

Fig. 13 shows sample qualitative results of the four methods. Results of SVMC, CF and CT are visualized as follows. For each clustered feature vector, we draw a colored circle on its location, and the color is assigned based on the mean of the orientations of the features inside the same cluster; this allows us to highlight regions with people having the same motion properties. Results for Streakflow are visualized as in the original paper [2].

In *Marathon*, SVMC clusters the four motion clusters formed by the runners (Fig. 13d), whereas CT detects only three motion clusters in the scene as it considers runners on the left-hand side and center having the same motion direction (Fig. 13a). Streakflow and CF cluster the whole flow of people as a single motion cluster without taking into consideration the motion direction of the runners.

In both *Traffic-junction* cases, SVMC clusters the two main motion clusters belonging to the two main lanes (dark red and cyan) while also distinguishing objects traversing the junction orthogonally to these lanes (yellow). Note that, the other methods cannot detect these two objects.

In *Train-station*, SVMC generates clusters that can be used to infer the main directions which people are heading in and to identify people moving in different directions within groups (dark red cluster on the left-hand side). We can observe that flows of people following in different directions are homogeneously clustered with SVMC, whereas with the other methods some people within the same flow are divided into sub-clusters (e.g. in the case of CT and CF where the flow on the right-hand side is split into blue and cyan clusters).

In *Student003 1* CF correctly clusters a group of people with a dark red cluster on the left-hand side, whereas SVMC considers the feature vectors in the area as outliers. In *Student003 2*, SVMC separately clusters groups of people walking close to each other (cyan and purple patches in the center of the image) but moving in the opposite direction, whereas Streakflow erroneously considers these people as having the same motion direction (yellow patch in the center). CF and CT generate motion clusters of a few groups of people while discarding other people in the scene.

VII. Conclusion

We proposed a novel unsupervised method for clustering motion vectors extracted from crowd scenes. This is achieved via the introduction of an on-line clustering performance evaluation measure (Dissimilarity Score) that provides feedback to SVC to tune the coarseness parameter in the kernel. A further refinement of the cluster boundaries is performed via the Quasiconformal Kernel Transformation from the Support

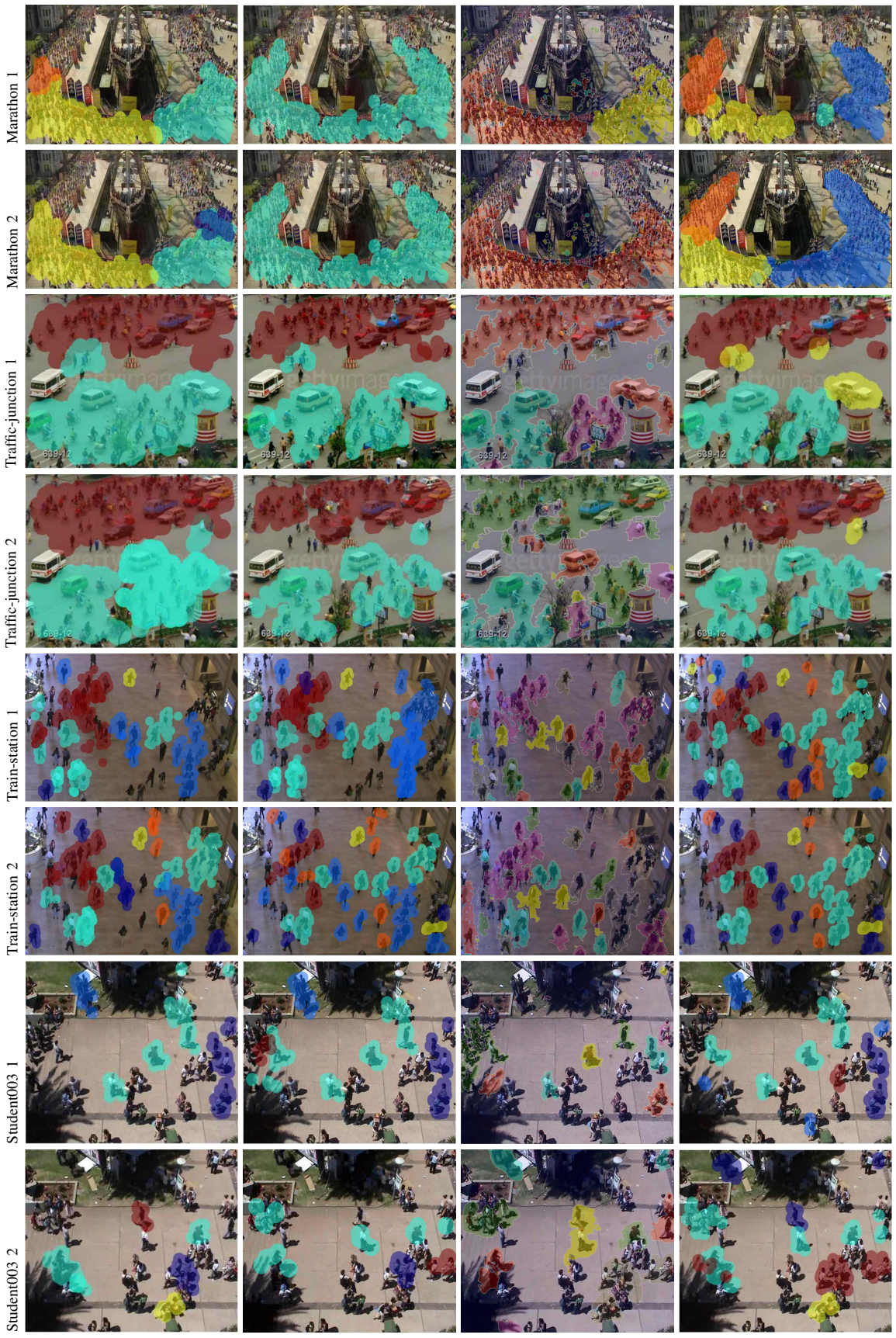


Fig. 13. Comparison of the crowd motion clustering results on different frames. Color patches represent motion clusters with the same motion direction (a) CT [12], (b) CF [4], (c) Streakflow [2], (d) SVMC.

Vector Machine framework. Temporal adaptation of the cluster model is carried out with incremental learning that considers the concept of obsolescence of feature vectors in order to keep the model updated only to current scene dynamics.

We validated the performance of the proposed method on an extended version of a state-of-the-art synthetic dataset [4] and showed that our clustering performance outperforms that of traditional clustering methods. Moreover, we applied our method to real-world crowd videos and showed that it effectively characterizes the flows of moving people.

In addition to the applications shown in this paper, our method could be used for group detection and tracking in crowd without explicitly tracking individuals [51]. A limitation of the proposed method is the computational cost due to the need to solve a quadratic problem. Future research will therefore include a reduction in the computational complexity by using a second-order Sequential Minimization Optimization [52] and by performing cluster labeling with a recent method proposed in [53].

REFERENCES

[1] M. Hu, S. Ali, and M. Shah, "Learning motion patterns in crowded scenes using motion flow field," in *Proc. of International Conference on Pattern Recognition*, Tampa, FL, USA, Dec. 2008, pp. 1–5.

[2] R. Mehran, B. Moore, and M. Shah, "A streakline representation of flow in crowded scenes," in *Proc. of European Conference on Computer Vision*, Crete, GR, Sep. 2010, pp. 439–452.

[3] S. Wu and H. Wong, "Crowd motion partitioning in a scattered motion field," *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, vol. 42, no. 5, pp. 1443–1454, Oct. 2012.

[4] B. Zhou, X. Tang, and X. Wang, "Coherent filtering: Detecting coherent motions from crowd clutters," in *Proc. of European Conference on Computer Vision*, Firenze, IT, Oct. 2012, pp. 857–871.

[5] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan, "Crowded scene analysis: A survey," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 367–386, Mar. 2015.

[6] S. Ali and M. Shah, "A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis," in *Proc. of Computer Vision and Pattern Recognition*, Minneapolis, MN, USA, Jun. 2007, pp. 1–6.

[7] L. Kratz and K. Nishino, "Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 987–1002, May 2012.

[8] S. Yi, H. Li, and X. Wang, "Understanding pedestrian behaviors from stationary crowd groups," in *Proc. of Computer Vision and Pattern Recognition*, Boston, MA, USA, Jun. 2015, pp. 3488–3496.

[9] S. Gong, M. Cristani, S. Yan, and C. Loy, Eds., *Person Re-Identification*. Springer, 2014.

[10] M. van de Vel, *Theory of convex structures*. North-Holland, 1993.

[11] H. Ullah, M. Ullah, and N. Conci, "Dominant motion analysis in regular and irregular crowd scenes," in *Proc. of Human Behavior Understanding Workshop*, Zurich, CH, Sep. 2014, pp. 62–72.

[12] S. Jing, C. Loy, and W. Xiaogang, "Scene-independent group profiling in crowd," in *Proc. of Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 2227–2234.

[13] B. Zhou, X. Wang, and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents," in *Proc. of Computer Vision and Pattern Recognition*, Providence, RI, USA, Jun. 2012, pp. 2871–2878.

[14] J. Shao, K. Kang, C. Loy, and X. Wang, "Deeply learned attributes for crowded scene understanding," in *Proc. of Computer Vision and Pattern Recognition*, Boston, MA, USA, Jun. 2015, pp. 4657–4666.

[15] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[16] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, Dec. 2001.

[17] C. Burges and D. Crisp, "Uniqueness of the SVM solution," in *Proc. of Neural Information Processing Systems*, Denver, CO, USA, Nov. 1999, pp. 223–229.

[18] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, Jul. 1999.

[19] M. Bicego, M. Cristani, A. Fusiello, and V. Murino, "Watershed-based unsupervised clustering," in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, ser. Lecture Notes in Computer Science, A. Rangarajan, M. Figueiredo, and J. Zerubia, Eds. Springer Berlin Heidelberg, 2003, vol. 2683, pp. 83–94.

[20] W. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, no. 1, pp. 7–24, Jan. 1984.

[21] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[22] X. Zhang, W. Wang, K. Nrvag, and M. Sebag, "K-AP: Generating specified K clusters by efficient affinity propagation," in *Proc. of International Conference on Data Mining*, Sydney, AU, Dec. 2010, pp. 1187–1192.

[23] T. K. D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient K-means clustering algorithm: analysis and implementation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 24, pp. 881–892, Jul. 2002.

[24] D. Comaniciu and P. Meer, "Mean Shift: A robust approach toward feature space analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002.

[25] W. Hu, X. Li, G. Tian, S. Maybank, and Z. Zhang, "An incremental DPMM-based method for trajectory clustering, modeling, and retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1051–1065, May 2013.

[26] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 176–190, Jan. 2008.

[27] G. Eibl and N. Brandle, "Evaluation of clustering methods for finding dominant optical flow fields in crowded scenes," in *Proc. of Pattern Recognition*, Tampla, FL, USA, Dec. 2008, pp. 1–4.

[28] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, Mar. 2003.

[29] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, Dec. 1971.

[30] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, Sep. 1974.

[31] D. Tax and R. Duin, "Support vector domain description," *Pattern Recognition Letters*, vol. 20, no. 11-13, pp. 1191–1199, Nov. 1999.

[32] C.-D. Wang and J. Lai, "Position regularized support vector domain description," *Pattern Recognition*, vol. 46, no. 3, pp. 875–884, Mar. 2013.

[33] P. Wolfe, "A duality theorem for non-linear programming," *Quarterly of Applied Mathematics*, vol. 12, no. 3, pp. 239–244, Aug. 1961.

[34] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computing*, vol. 13, no. 3, pp. 637–649, Mar. 2001.

[35] J.-S. Wang and J.-C. Chiang, "A cluster validity measure with a hybrid parameter search method for the support vector clustering algorithm," *Pattern Recognition*, vol. 41, no. 2, pp. 506–520, Feb. 2008.

[36] K.-P. Wu and S.-D. Wang, "Choosing the kernel parameters for support vector machines by the inter-cluster distance in the feature space," *Pattern Recognition*, vol. 42, no. 5, pp. 710–717, May 2009.

[37] S.-H. Lee and K. Daniels, "Gaussian kernel width generator for support vector," in *Proc. of Bioinformatics and its Applications*, Singapore, May 2005, pp. 151–162.

[38] D. Anguita, A. Ghio, I. Lawal, and L. Oneto, "A heuristic approach to model selection for online support vector machines," in *Proc. of Advances in Regularization, Optimization, Kernel Methods and Support Vector Machines (Workshop)*, Leuven, BE, Jul 2013, pp. 77–78.

[39] P. Berens, "CircStat: A MATLAB toolbox for circular statistics," *Journal of Statistical Software*, vol. 31, no. 10, Sep. 2009.

[40] J.-S. Wang and J.-C. Chiang, "A cluster validity measure with a hybrid parameter search method for the support vector clustering algorithm," *Pattern Recognition*, vol. 41, no. 2, pp. 506–520, 2008.

[41] S.-H. Lee and K. M. Daniels, "Gaussian kernel width exploration and cone cluster labeling for support vector clustering," *Pattern Analysis and Application*, vol. 15, no. 3, pp. 327–344, Aug. 2012.

[42] E. W. Weisstein. Piecewise constant function. MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/PiecewiseConstantFunction.html
[43] J. Peng, D. R. Heisterkamp, and H. K. Dai, "Adaptive Quasiconformal kernel nearest neighbor classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 656–661, May 2004.
[44] S.-H. Lee and K. M. Daniels, "Cone cluster labeling for support vector clustering," in *Proc. of Data Mining*, Bethesda, MD, USA, May 2006, pp. 484–488.
[45] R. Gomes, M. Welling, and P. Perona, "Incremental learning of nonparametric Bayesian mixture models," in *Proc. of Computer Vision and Pattern Recognition*, Anchorage, AK, USA, Jun. 2008, pp. 1–8.
[46] C.-D. Wang, J.-H. Lai, and D. Huang, "Incremental Support Vector Clustering," in *Proc. of the International Conference on Data Mining Workshops*, Vancouver, CA, Dec. 2011, pp. 839–846.
[47] L. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. MIT Press, Cambridge, MA., 2007, pp. 1–28.
[48] J. Lee and D. Lee, "An improved cluster labeling method for support vector clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 461–464, Mar. 2005.
[49] ——, "Dynamic characterization of cluster structures for robust and inductive support vector clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1869–1874, Nov. 2006.
[50] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1644–1656, Aug. 2014.
[51] R. Mazzon, F. Poiesi, and A. Cavallaro, "Detection and tracking of groups in crowd," in *Proc. of Advanced Video and Signal-Based Surveillance*, Krakow, PL, Aug. 2013, pp. 202–207.
[52] T. Glasmachers and C. Igel, "Second-order SMO improves SVM online and active learning," *Neural Computation*, vol. 20, no. 2, pp. 374–382, Jan. 2008.
[53] Y. Ping, Y. Chang, Y. Zhou, Y. Tian, Y. Yang, and Z. Zhang, "Fast and scalable support vector clustering for large-scale data analysis," *Journal of Knowledge and Information Systems*, vol. 43, no. 2, pp. 281–310, May 2015.

Isah A. Lawal received the B.Eng. degree in Electrical and Electronic Engineering in 2002 from Abubakar Tafawa Balewa University, Nigeria, the M.Sc. degree in Computer Engineering in 2010 from King Fahd University of Petroleum and Minerals, Saudi Arabia, and the Ph.D. degree in Electronic Engineering and Computer Science from University of Genoa, Italy, and Queen Mary University of London, UK. He is currently a Lecturer in Artificial Intelligence at the department of Computer Science, Federal University Dutse, Nigeria. His research focuses on incremental support vector-based characterization of feature flows and machine learning applications for video analysis.

Fabio Poiesi received the B.Sc. and M.Sc. degrees in telecommunication engineering from University of Brescia, Italy, in 2007 and 2010, respectively, and the Ph.D. degree in electronic engineering and computer science from Queen Mary University of London, U.K., in 2014. He is a Post-Doctoral Research Assistant with Queen Mary University of London, with Prof. A. Cavallaro under the European project ARTEMIS Cognitive and Perceptive Cameras. His research interests include video multiobject detection and tracking in highly populated scenes, performance evaluation of tracking algorithms, and behavior understanding for the analysis of human interactions in crowds.

Davide Anguita (SM'12) received the Laurea degree in electronic engineering and the Ph.D. in computer science and electronic engineering from University of Genoa, Genoa, Italy, in 1989 and 1993, respectively. He was a Research Associate with the International Computer Science Institute, Berkeley, CA, USA, working on special-purpose processors for neurocomputing, and then returned to the University of Genoa, where he is currently Associate Professor of Computer Engineering with the Department of Informatics, Bioengineering, Robotics, and Systems Engineering. His current research interests focus on theory and applications of kernel methods and artificial neural networks.

Andrea Cavallaro received the Ph.D. degree in electrical engineering from Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2002. He was a Research Fellow with BT Group PLC, London, U.K., in 2004. He is currently a Professor of Multimedia Signal Processing and the Director of the Centre for Intelligent Sensing at Queen Mary University of London, London. He has authored over 160 journal and conference papers, one monograph on video tracking (Wiley, 2011), and three edited books entitled Multi-Camera Networks (Elsevier, 2009), Analysis, Retrieval and Delivery of Multimedia Content (Springer, 2012), and Intelligent Multimedia Surveillance (Springer, 2013). Dr. Cavallaro received the Royal Academy of Engineering Teaching Prize in 2007; three student paper awards on target tracking and perceptually sensitive coding at the IEEE International Conference on Acoustics, Speech, and Signal Processing in 2005, 2007, and 2009; and the best paper award at the IEEE Advanced Video and Signal Based Surveillance Conference in 2009. He is an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and a member of the Editorial Board of IEEE Multimedia. He was an Area Editor of IEEE Signal Processing Magazine and an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON SIGNAL PROCESSING, and IEEE Signal Processing Magazine.