Computer Vision and Image Understanding 117 (2013) 1–11

Contents lists available at SciVerse ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Modeling dynamic swarms ☆

Bernard Ghanem [a,*], Narendra Ahuja [b]

[a] King Abdullah University of Science and Technology, Geometric Modeling and Scientific Visualization Center, Saudi Arabia
[b] University of Illinois at Urbana-Champaign, Electrical and Computer Engineering Department, USA

ARTICLE INFO

Article history:
Received 28 October 2011
Accepted 13 September 2012
Available online 26 September 2012

Keywords:
Swarms
Dynamic textures
Crowd behavior analysis
Spatiotemporal analysis
Computer vision

ABSTRACT

This paper proposes the problem of modeling video sequences of dynamic swarms (DSs). We define a DS as a large layout of stochastically repetitive spatial configurations of dynamic objects (swarm elements) whose motions exhibit local spatiotemporal interdependency and stationarity, i.e., the motions are similar in any small spatiotemporal neighborhood. Examples of DS abound in nature, e.g., herds of animals and flocks of birds. To capture the local spatiotemporal properties of the DS, we present a probabilistic model that learns both the spatial layout of swarm elements (based on low-level image segmentation) and their joint dynamics that are modeled as linear transformations. To this end, a spatiotemporal neighborhood is associated with each swarm element, in which local stationarity is enforced both spatially and temporally. We assume that the prior on the swarm dynamics is distributed according to an MRF in both space and time. Embedding this model in a MAP framework, we iterate between learning the spatial layout of the swarm and its dynamics. We learn the swarm transformations using ICM, which iterates between estimating these transformations and updating their distribution in the spatiotemporal neighborhoods. We demonstrate the validity of our method by conducting experiments on real and synthetic video sequences. Real sequences of birds, geese, robot swarms, and pedestrians evaluate the applicability of our model to real world data.

1. Introduction

This paper is about modeling video sequences of a dense collection of moving objects which we will call swarms. Examples of dynamic swarms (DSs) in nature abound: a colony of ants, a herd of animals, people in a crowd, a flock of birds, a school of fish, a swarm of honeybees, trees in a storm, and snowfall. In artificial settings, dynamic swarms are illustrated by: fireworks, a caravan of vehicles, sailboats on a lake, and robot swarms. A DS is characterized by the following properties. (1) All swarm elements belong to the same category. This means that the appearances (i.e. geometric and photometric properties) of the elements are similar although not identical. For example, each element may be a sample from the same underlying probability density function (pdf) of appearance parameters. (2) The swarm elements occur in a dense spatial configuration. Thus, their spatial placement, although not regular, is statistically uniform, e.g., determined by a certain pdf. (3) Element motions are statistically similar. (4) The motions of the swarm elements are globally independent. In other words, the motions of two elements that are sufficiently well separated are independent. However, this is not strictly true on a local scale because if they are located too close compared to the extents of their displacements, then their motions must be interdependent to preserve separation. Thus, the motion parameters of each element vs. the other elements can be considered as being chosen from a mutually conditional pdf. Occasional variations in these swarm properties are also possible, e.g. elements may belong to multiple categories such as different types of vehicles in traffic. Fig. 1 shows some examples of DS.

This definition of DS is reminiscent of dynamic textures (DT). Indeed, a DS is analogous to a DT of complex nonpoint objects. The introduction of complex nonpoint objects introduces significant complexity: (1) Extraction of nonpoint objects becomes necessary, whose added complexity is evident from, e.g., the algorithm of [1]. (2) Motion for nonpoint objects is richer than point objects, e.g., rotation and nonrigid transformations become feasible. Since most work on DTs has focused on textures formed of pixel or subpixel objects, DS is a relatively unexplored problem. Tools for DS analysis should be useful for general problems such as dynamic scene recognition, dynamic scene synthesis, and anomaly detection, as well as, specific problems such as the motion analysis of animal herds or flocks of birds. In this paper, we present an approach to derive the model of a DS from its video, and demonstrate its efficacy through example applications. Before we do this, we first review the work most related to DS, namely, that on DT.

☆ This paper has been recommended for acceptance by Y. Aloimonos.

* Corresponding author.
E-mail addresses: bernard.ghanem@kaust.edu.sa (B. Ghanem), n-ahuja@illinois.edu (N. Ahuja).

1077-3142/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2012.09.002

Fig. 1. Examples of swarms.

1.1. Related work

A DT sequence captures a random spatiotemporal phenomenon which may be the result of a variety of physical processes, e.g., involving objects that are small (smoke particles) or large (snowflakes), or rigid (flag) or nonrigid (cloud, fire), moving in 2D or 3D, etc. Even though the overall global motion of a DT may be perceived by humans as being simple and coherent, the underlying local motion is governed by a complex stochastic model. Irrespective of the nature of the physical phenomena, the objective of DT modeling in computer vision and graphics is to capture the nondeterministic, spatial and temporal variation in images.

As discussed earlier, although the basic notion of DTs allows that both spatial and temporal variations be complex, the limited work done on DT's has focused on moving objects (texels) that have little spatial complexity, even as they exhibit complex motion. The texels are of negligible size (e.g. smoke particles), whose movement appears as a continuous photometric variation in the image, rather than as a sparser arrangement of finite (nonzero) size texels. Consequently, the DT model must mainly capture the motion and less is needed to represent the spatial structure.

Statistical modeling of spatiotemporal interdependence among DT images serves as being closest to the work we present here. This work includes the spatiotemporal auto-regressive (STAR) model by Szummer and Picard [2] and multi-resolution analysis (MRA) trees by Bar-Joseph et al. [3]. The DT model of Soatto et al. [4] uses a stable linear dynamical system (LDS). LDS mixture models have been developed in [5] and applied to DT clustering and segmentation. A bag-of-LDS model is proposed in [6] to account for view-invariance in DT recognition. Furthermore, the basic LDS model is extended to represent the incidence of multiple co-occurring DTs in the same video sequence, thus, leading to a layered LDS model for video [7,8]. In [9], a mixture of globally coordinated PPCA models is employed to model a DT. Moreover, a DT can be represented as a distribution of responses to spatiotemporal filters encoding oriented structures, which are shown to be discriminative of different DT classes [10]. Recently, the spatiotemporal variations in a DT has been described using dynamic fractal analysis, which in turn has shown great success in DT classification [11].

Along with their merits, the previously proposed models also suffer from certain shortcomings. (i) These models make restrictive assumptions about the DT sequences. Most of them assume that there is a single DT covering each frame in the sequence, while the others that consider multiple DT's are usually limited to particle textures (e.g. water and smoke). Consequently, these models cannot be easily extended to dynamic swarms. Even if the texels were known beforehand, learning a separate model for each texel does not guarantee the underlying spatiotemporal stationarity of DS. (ii) They do not make a clear separation between the appearance and dynamical models of the DT. The approach proposed in [12] explicitly aims at this separation, but it is limited to fluid DT's only.

Another body of work that is related to our swarm motion models a DT as a set of dynamic *textons* (or *motons*) whose motion is governed by a Markov chain model [13,14]. This generative model is limited to sequences of particle objects (e.g. snowflakes) or objects imaged at large distances. The texton dynamics are constrained by the underlying assumptions of the model, which state that all textons have the same frame-to-frame transformation, that this transformation is constant over time, and that the dynamics of spatially neighboring textons are independent. While this work does involve moving objects containing more than one pixel per object as well as some interpixel spacing, its modeling power still does not match the needs of properties (1–4) of a DS given above.

In the rest of this paper, we refer to the objects forming a swarm as swarm *elements*. We propose a probabilistic model that learns both the spatial layout of the swarm elements and their joint dynamics, modeled as linear transformations, which allow for a clear separation between the appearance and dynamics of these elements. This joint representation takes into account the interdependence in the properties of elements that are neighbors in space and time. This is done by enforcing stationarity only within spatiotemporal neighborhoods. This local stationarity constraint allows us to model DS sequences that not only exhibit globally uniform dynamics (to which previous methods are limited), but also sequences whose element properties and dynamics gradually change, in space and time.

1.2. Overview of proposed model

Given a DS sequence in which swarm elements undergo locally stationary transformations, we iterate between learning the spatial layout of these elements (their binary alpha mattes and frame-to-frame correspondences) and their dynamics. We estimate swarm dynamics such that they follow a probabilistic model that enforces

local stationarity within a spatiotemporal neighborhood of each element. In regards to spatial layout, we assume that each swarm element consists of one or more homogenous segments that also possess these spatiotemporal stationarity properties.

We model the frame-to-frame motion of each individual element as a linear transformation, which reconstructs the element's features in a given frame from its features in the previous one. These features can describe local or global properties. In our framework, we do not restrict the choice of these features, since they can be application dependent. These linear transformations are chosen to capture a wide variety of possible changes especially rotation, scaling, and shear. Moreover, a spatiotemporal neighborhood is associated with each element, in which local stationarity is enforced. Spatially, this is done by assuming that the dynamics of elements in a given neighborhood are samples from the same distribution corrupted by i.i.d. Gaussian noise. Temporally, these dynamics are governed by an autoregressive (AR) model. We learn swarm dynamics by estimating the transformations that maximize the a posteriori probability or equivalently that (i) minimize the reconstruction error and (ii) enforce stationarity in each element's neighborhood.

Contributions: (1) We present an approach that learns the dynamics of swarm elements jointly. This is done by modeling their frame-to-frame linear transformations instead of directly modeling their features. Using these transformations, our model is able to handle more complex swarm motions and allows for a clear separation between the appearance and dynamics of a swarm. (2) Based on our assumption of local spatiotemporal stationarity, the proposed probabilistic model allows for interdependence between swarm elements both in time and space. This is done locally, so as not to limit the types of DS sequences that can be modeled, which is a shortcoming of most other methods. (3) The proposed model and learning algorithm estimate the spatial layout of swarm elements by enforcing temporal coherence in determining their frame-to-frame correspondences and the spatial stationarity of their dynamics.

The paper is organized as follows. In Section 2, we give a detailed description of our proposed probabilistic model and the learning algorithm that estimates its optimal parameters. Section 3 evaluates the performance of our model on synthetic and real world data with applications to action recognition and motion segmentation.

2. Proposed spatiotemporal model

In this section, we give a detailed description of our spatiotemporal model for the spatial layout and dynamics of a DS. We consider sequences whose fundamental spatial elements are opaque objects. The changes these elements undergo are stationary, both spatially and temporally. We also assume that each swarm element consists of one or more homogenous segments that also possess these spatiotemporal stationarity properties. To learn the spatial layout of a swarm, we refrain from using texel extraction algorithms (e.g [1]) from the literature because they do not make complete use of the spatiotemporal relationship inherent to swarm elements. We also avoid the use of popular multiple object trackers in the literature (e.g. [15]), since they require either manual initialization or class-specific information. If prior information on the DS elements in the scene is known, it can be incorporated into our proposed framework to facilitate the learning of the DS spatial layout. We revisit the video segmentation algorithm of [16], which has some interesting properties that we exploit to learn spatial layout. Since no explicit tracking is performed on the swarm elements, occlusion handling remains a problem and is left for future work. To enforce stationarity, we assume that the dynamics of the swarm elements are distributed according to an MRF in both space and time. In our model, the dynamics of each swarm element is influenced by its spatial and temporal neighbors, within its spatiotemporal neighborhood. Unlike other dynamical models (e.g. [4,14]) that assume spatial independence between texture elements, we maintain spatiotemporal dependence among swarm elements to render a more constrained model. In what follows, we give a clear mathematical formulation of our problem.

2.1. Overview of the DS framework

We are given F frames of size $M \times N$ constituting a swarm sequence. Frame t in this sequence contains K_t swarm elements. This allows elements to disappear and re-appear at different time instances. We define a swarm element as similar-looking objects that also have similar dynamics. In the absence of a general object segmentation algorithm, a swarm element is modeled in our DS framework as one or more adjacent low-level image segments that have share similar dynamical properties. Although any low-level segmentation algorithm can ideally be used here, we choose the algorithm in [17] due to the reasons stated in Section 2.1.1. Section 2.1.2 summarizes the mathematical notation used in the rest of this paper. In the following sections, we show how we iterate between learning the spatial layout of the swarm elements and their dynamics. At a given iteration, we fix element dynamics and update the swarm elements by clustering segments to enforce spatiotemporal stationarity. Then, we update the dynamics of the new swarm elements.

2.1.1. Hierarchical low-level image segmentation

In general, low-level image segmentation partitions a given image into regions which are characterized by some low-level properties of their constituent pixels, where the term "low-level" refers to local and intrinsic properties of pixels such as gray-level intensity (or color), contrast, and gradient. To represent dynamic swarm elements, we use hierarchical low-level image segmentation, namely the low-level multi-scale (LLMS) segmentation algorithm proposed in [17]. Note that if a reliable object segmentation method is available, it can be used in lieu of LLMS.

Since the mechanism used by the human visual system (HVS) to produce perceptual groupings of objects in video remains unknown, we do not claim that LLMS (or any other low-level segmentation algorithm) is capable of reliably generating segmentations of objects in video. However, we do assume that LLMS can produce a hierarchy, whose constituents may play an important role in forming dynamic objects. In fact, there is evidence that the HVS initially forms groupings of pixels that have a certain degree of interior homogeneity and a discontinuity with their surroundings, thus, stressing the significance of contrast. It is this low-level intensity contrast that LLMS uses to define and detect image segments. In the LLMS approach, a segment is modeled as a connected set of pixels whose interior variation is inferior to the magnitude of the discontinuity with their surroundings. The fact that a segment can have an arbitrary shape and size precludes the use of any prior model about its geometry. Moreover, since it is unclear what degree (s) of interior homogeneity are acceptable for object formation, multiscale analysis is necessary. LLMS guarantees a strict photometric hierarchy with recursive containment relations. The levels of the hierarchy are chosen from the range of possible contrast levels allowed for grouping connected pixels. The main reasons for choosing LLMS over other low-level algorithms are that it does not require significant user parameters and it outperforms popularly used methods on a low-level human segmentation benchmark [17]. In our experiments, the dynamic segments[1]

[1] We do not consider segments that remain static for elongated periods of time.

produced by the LLMS hierarchy tend to correspond to object parts or entire objects. In this work, we use dynamic segments as the representative building blocks for swarm elements. Estimating the spatial layout of an element in a given frame is equivalent to determining the subset of these dynamic segments that constitute the element in this frame. This estimation will be described in detail in the following sections.

2.1.2. Mathematical notation

Let us denote the swarm elements by their spatial layouts (i.e. binary alpha mattes) $\{\mathbf{T}_t^{(i)} : t = 1,\dots,F; i = 1,\dots,K_t\}$, where $\mathbf{T}_t^{(i)}$ is the manifestation or spatial layout[2] of the i^{th} element in frame t and $\mathbb{T}_t = \{\mathbf{T}_t^{(i)} : i = 1,\dots,K_t\}$ is the set of swarm elements in frame t. Swarm element $\mathbf{T}_t^{(i)}$ is represented by its d-dimensional feature vector $\vec{\mathbf{f}}_t^{(i)}$, which can describe its appearance, shape, and spatial layout. To model local swarm dynamics, we define a linear transformation $\mathbf{A}_t^{(i)}$ that transforms $\vec{\mathbf{f}}_t^{(i)}$ into $\vec{\mathbf{f}}_{t+1}^{(i)}$. Due to its general form, it can encompass commonly used transformations (e.g. rotation and scaling) as well as more specific ones (e.g. any orthogonal or orthonormal transformation). We use $\mathbb{A}_t = \{\mathbf{A}_t^{(i)} : i = 1,\dots,K_t\}$ to denote the set of transformations for the K elements in frame t and $\mathbb{F}_t = \{\vec{\mathbf{f}}_t^{(i)} : i = 1,\dots,K_t\}$ to denote the set of features.

By using frame-to-frame transformations to characterize swarm dynamics instead of their corresponding features, we emphasize the separation between swarm appearance and dynamics. This is usually ignored in other models. This explicit separation allows distinction between and independent control of elements' appearance and motion. That is, we can pair any swarm elements with any dynamics. The goals of modeling these linear transformations are twofold.

- **[G1]** We desire accurate frame-to-frame reconstruction of the feature vectors, which determines how well our model fits the underlying data.
- **[G2]** We need to impose spatial and temporal stationarity on the transformations within a local spatiotemporal neighborhood. In the absence of **[G2]**, our model is ill-posed and too general for any practical use. Consequently, **[G2]** ensures that our model conforms to the underlying process that generates the swarm elements' dynamics.In what follows, Section 2.2 gives a detailed description of how a swarm element's spatiotemporal neighborhood is formed. In Section 2.4, we learn the spatial layout and the linear transformations in a probabilistic MAP framework.

2.2. Spatiotemporal neighborhood in a DS

Our dynamical model assumes spatial and temporal stationarity for each swarm element within its spatiotemporal neighborhood. Let $\mathscr{C} = \{\mathbf{N}_t^{(i)} : t = 1,\dots,F; i = 1,\dots,K_t\}$ be the set of all spatiotemporal neighborhoods in the sequence. $\mathbf{N}_t^{(i)}$ is the set of elements included in the neighborhood of $\mathbf{T}_t^{(i)}$. We define $\Gamma(t,i)$ to be the set of index pairs (u,v) that represent $\mathbf{T}_u^{(v)}$ in $\mathbf{N}_t^{(i)}$. For simplicity, we decompose $\Gamma(t,i)$ into two disjoint sets of indices, $\Gamma_S(t,i)$ and $\Gamma_T(t,i)$, where $\Gamma_S(t,i) = \{(t,j) : \mathbf{T}_t^{(j)} \in \mathbf{N}_t^{(i)}\}$ and $\Gamma_T(t,i) = \{(s,i) : \mathbf{T}_s^{(i)} \in \mathbf{N}_t^{(i)}\}$. $\Gamma_S(t,i)$ defines the spatial neighbors of $\mathbf{T}_t^{(i)}$, while $\Gamma_T(t,i)$ defines its temporal neighbors.

2.2.1. Spatial neighborhood

The elements, indexed by $\Gamma_S(t,i)$, are determined by the generalized Voronoi regions [18] corresponding to the elements present in the t^{th} frame. We also weigh the "neighborness" of every pair of spatial neighbors. $w_t(i,j)$ is the corresponding weight of the elements pair $\left(\mathbf{T}_t^{(i)}, \mathbf{T}_t^{(j)}\right)$. It is equal to the ratio of the length of the common boundary between the Voronoi regions of the neighboring elements, to the average distance of these elements to the common boundary. For elements that are not spatial neighbors, this weight is set to zero. Local spatial stationarity is enforced by assuming that transformations of neighboring elements are drawn from the same distribution, corrupted by Gaussian i.i.d. noise. Therefore, we have

$$\forall (t,j) \in \Gamma_S(t,i) : \mathbf{A}_t^{(j)} = \mathbf{A}_t^{(i)} + \mathbf{E}, \quad \text{such that}$$
$$\mathbf{E}(u,v) \sim \mathscr{N}\left(0, \frac{\sigma_S^2}{w_t(i,j) + \varepsilon}\right) \forall u, \quad v = 1,\cdots,d \tag{1}$$

2.2.2. Temporal neighborhood

The elements, indexed by $\Gamma_T(t,i)$, are the manifestations of the ith element in a temporal window consisting of the W_T previous frames. The limits of this window are truncated to remain within the limits of the video sequence itself. This is done to resolve exceptions occurring at the first W_T frames in the sequence. We enforce temporal stationarity by applying an AR model of order W_T to the sequence of transformations in this temporal window. In fact, the AR model has often been used to model features over time [13], but here, we use it to model the temporal variations of these features (i.e. the dynamics themselves). Therefore, we have

$$\forall \mathbf{N}_t^{(i)} \in \mathscr{C} : \mathbf{A}_t^{(i)} = \sum_{j=1}^{\rho_t} \alpha_j \mathbf{A}_{t-j}^{(i)} + \mathbf{E}, \quad \text{such that}$$
$$\mathbf{E}(u,v) \sim \mathscr{N}(0,\sigma_T^2) \text{ and } \rho_t = \min(W_T, t-1) \tag{2}$$

For simplicity, the AR coefficients ($\vec{\alpha} \in \mathbb{R}^{W_T}$), are assumed to be time invariant and constant for all swarm elements. In Fig. 2, we show an example of the spatiotemporal neighborhood of $\mathbf{T}_t^{(1)}$ with $W_T = 2$. Note that the number of spatial neighbors and the "neighborness" weights can change from frame-to-frame.

2.3. Model of swarm dynamics and spatial layout

Here, we present the probabilistic model that governs the dynamics of swarm elements and their spatial layout in a DS. We model the joint probability of the spatial layout of the swarm elements, their features, and their dynamics. This is done by decomposing the joint into the prior over the transformations and the spatial layout, in addition, to the likelihood of the features given the swarm layout and dynamics as in (3). In what follows, we model the three terms to ensure **[G1]** and **[G2]**.

$$p\left(\{\mathbb{A}_t\}_{t=1}^{F-1}, \{\mathbb{F}_t\}_{t=1}^{F-1}, \{\mathbb{T}_t\}_{t=1}^{F}\right) = \mathscr{L}\mathscr{P}_{\mathbb{T}}\mathscr{P}_{\mathbb{A}} \tag{3}$$

where

$$\mathscr{L} = p\left(\{\mathbb{F}_t\}_{t=1}^{F} | \{\mathbb{A}_t\}_{t=1}^{F-1}, \quad \{\mathbb{T}_t\}_{t=1}^{F}\right),$$
$$\mathscr{P}_{\mathbb{T}} = p\left(\{\mathbb{T}_t\}_{t=1}^{F} | \{\mathbb{A}_t\}_{t=1}^{F-1}\right), \quad \text{and } \mathscr{P}_{\mathbb{A}} = p\left(\{\mathbb{A}_t\}_{t=1}^{F-1}\right)$$

2.3.1. Likelihood model ($\mathscr{L}$)

Since we assume a linear relationship between consecutive feature vectors, we can decompose the likelihood probability as: $\mathscr{L} = p_1 \prod_{t=1}^{F-1} \prod_{i=1}^{K_t} p\left(\vec{\mathbf{f}}_{t+1}^{(i)} | \vec{\mathbf{f}}_t^{(i)}, \mathbf{A}_t^{(i)}, \mathbb{T}_t\right)$, where $p\left(\vec{\mathbf{f}}_{t+1}^{(i)} | \vec{\mathbf{f}}_t^{(i)}, \mathbf{A}_t^{(i)}, \mathbb{T}_t\right) = \mathscr{N}\left(\mathbf{A}_t^{(i)}\vec{\mathbf{f}}_t^{(i)}, \gamma_t^2 I_d\right)$ and $p_1 = p\left(\mathbb{F}_1 | \{\mathbb{A}_t\}_{t=1}^{F-1}, \{\mathbb{T}_t\}_{t=1}^{F}\right)$ is a constant with respect to the transformations. Consequently, we can write the negative log likelihood as in (4).

[2] The spatial layout of an element is the union of the pixels comprising its constituent low-level segments.

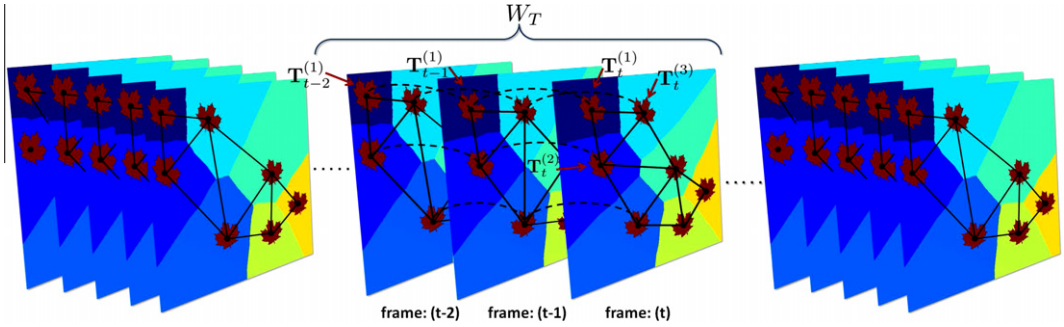


Fig. 2. Spatial neighbors are connected by solid black lines, while temporal neighbors are connected by dashed black lines. Here, $\mathbf{T}_t^{(1)}$ has two spatial neighbors ($\mathbf{T}_t^{(2)}$ and $\mathbf{T}_t^{(3)}$) and two temporal neighbors ($\mathbf{T}_{t-1}^{(1)}$ and $\mathbf{T}_{t-2}^{(1)}$) comprising its spatiotemporal neighborhood.

$$-\ln(\mathscr{L}) = \sum_{t=1}^{F-1}\left(\frac{dK_t}{2}\ln(\gamma_t^2) + \frac{1}{\gamma_t^2}\sum_{i=1}^{K_t}\left\|\vec{\mathbf{f}}_{t+1}^{(i)} - \mathbf{A}_t^{(i)}\vec{\mathbf{f}}_t^{(i)}\right\|_2^2\right) - \ln(p_1) + \frac{d\ln(2\pi)}{2}\sum_{t=1}^{F-1}K_t \tag{4}$$

2.3.2. Prior of swarm spatial layout ($\mathscr{P}_{\mathbb{T}}$)

As stated before, each swarm element consists of one or more homogenous segments that are produced by the algorithm of [17]. The spatial layout of these elements and their frame-to-frame correspondences must ensure that the swarm elements' features are reconstructed faithfully and that spatial stationarity of their dynamics is enforced. The frame-to-frame correspondences of a swarm element are equivalent to many-to-many correspondences between segments from the two frames. To formalize this problem, we denote the frame-to-frame correspondence between $\mathbf{T}_t^{(i)}$ and $\mathbf{T}_{t+1}^{(i)}$ as $n_{(t,i)}$, which is a node in the graph of all frame-to-frame correspondences in the swarm sequence. Two nodes $n_{(t,i)}$ and $n_{(s,j)}$ are considered neighbors in the graph, if any pair of $\left(\mathbf{T}_t^{(i)}, \mathbf{T}_{t+1}^{(i)}\right)$ and $\left(\mathbf{T}_s^{(j)}, \mathbf{T}_{s+1}^{(j)}\right)$ are spatially adjacent (i.e. share boundaries). We show an example in Fig. 3.

Here, we can define a self-similarity function for each node, $s_1(n_{(t,i)})$, that quantifies the quality of frame-to-frame feature reconstruction. Also, we define a pairwise similarity function for each pair of neighboring nodes, $s_2(n_{(t,i)}, n_{(s,j)})$, that evaluates how similar their frame-to-frame transformations are. This setup is similar to the one used in [16]. Actually, we shall see later that we use a similar method to update the spatial layout. We use normalized correlation to define $s_1(.)$ and $s_2(.)$, where $s_1(n_{(t,i)}) = \frac{\vec{\mathbf{f}}_{t+1}^{(i)T}\mathbf{A}_t^{(i)}\vec{\mathbf{f}}_t^{(i)}}{\|\vec{\mathbf{f}}_{t+1}^{(i)}\|_2\|\mathbf{A}_t^{(i)}\vec{\mathbf{f}}_t^{(i)}\|_2}$ and $s_2(n_{(t,i)}, n_{(s,j)}) = \frac{\mathrm{trace}(\mathbf{A}_t^{(i)T}\mathbf{A}_s^{(j)})}{\|\mathbf{A}_t^{(i)}\|_F\|\mathbf{A}_s^{(j)}\|_F}$. The prior $\mathscr{P}_{\mathbb{T}}$ is proportional to the self and pairwise similarities of all neighboring nodes in the graph.

2.3.3. Prior of Swarm Dynamics ($\mathscr{P}_{\mathbb{A}}$)

As $\mathscr{L}$ was modeled to guarantee **[G1]**, **[G2]** is accounted for by modeling $\mathscr{P}_{\mathbb{A}}$ as a product of potential functions defined on the set of all spatiotemporal neighborhoods. This decomposition is widely used to model priors on maximum cliques defined on an undirected graph. We define the potential function for each clique as the product of a spatial potential $\Psi_S(.)$ and a temporal potential $\Psi_T(.)$, which guarantee spatial and temporal stationarity in swarm dynamics, respectively. So, we have $\mathscr{P}_{\mathbb{A}} = \frac{1}{Z}\prod_{\mathbf{N}_t^{(i)}\in\mathscr{C}}\left[\Psi_S\left(\mathbf{N}_t^{(i)}\right)\Psi_T\left(\mathbf{N}_t^{(i)}\right)\right]$, where

$$\Psi_S\left(\mathbf{N}_t^{(i)}\right) = \prod_{\{(t,j),(t,j')\}\subset\Gamma_S(t,i);j\neq j'}\phi_S\left(\mathbf{A}_t^{(j)}, \mathbf{A}_t^{(j')}\right) \text{ and } \Psi_T\left(\mathbf{N}_t^{(i)}\right) = \phi_T\left(\left\{\mathbf{A}_s^{(i)} : \mathbf{T}_s^{(i)} \in \mathbf{N}_t^{(i)}\right\}\right)$$

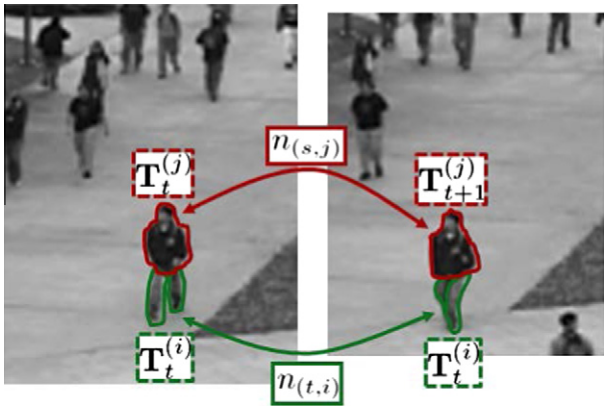


Fig. 3. Two neighboring nodes of swarm elements in frames t and $t+1$. Note that the $n_{(t,i)}$ consists of two regions.

$\phi_S(.)$ and $\phi_T(.)$ are potentials that evaluate how spatially and temporally stationary the swarm transformations are. For simplicity, we set $\phi_S(\mathbf{A}_t^{(j)}, \mathbf{A}_t^{(j')}) = p(\mathbf{A}_t^{(j)}|\mathbf{A}_t^{(j')})$ and $\Psi_T(\mathbf{N}_t^{(i)}) = p(\mathbf{A}_t^{(i)}|\{\mathbf{A}_{t-j}^{(i)}\}_{j=1}^{\rho_t})$. Using (1) and (2), we can express the negative log prior as in (5). Note that p_2 is a constant function of the "neighborness" weights, $C_S = \sum_{\mathbf{N}_t^{(i)}\in\mathscr{C}}\frac{d^2}{2}|\Gamma_S(t,i)|^2$, and $C_T = \sum_{\mathbf{N}_t^{(i)}\in\mathscr{C}}\frac{d^2}{2}|\Gamma_T(t,i)|^2$. Also, we assume that the normalizing factor Z is constant with respect to the swarm dynamics, the noise variances, and the AR coefficients. This simplifying assumption is commonly made in approaches that make use of graph potentials.

$$-\ln(\mathscr{P}_{\mathbb{A}}) = \ln(Z) + \ln(p_2) + C_S\ln(\sigma_S^2) + C_T\ln(\sigma_T^2) + \frac{1}{\sigma_S^2}\sum_{\mathbf{N}_t^{(i)}\in\mathscr{C}}\left(\sum_{\{(t,j),(t,j')\}\subset\Gamma_S(t,i);j\neq j'} w_t(j,j')\left\|\mathbf{A}_t^{(j)} - \mathbf{A}_t^{(j')}\right\|_F^2\right) + \frac{1}{\sigma_T^2}\sum_{\mathbf{N}_t^{(i)}\in\mathscr{C}}\left\|\mathbf{A}_t^{(i)} - \sum_{j=1}^{\rho_t}\alpha_j\mathbf{A}_{t-j}^{(i)}\right\|_F^2 \tag{5}$$

2.4. Learning swarm layout and dynamics

After establishing our probabilistic model, we proceed to learning its parameters, $\{\mathbb{T}_t\}_{t=1}^{F}, \{\mathbb{A}_t\}_{t=1}^{F-1}$, the noise variances σ_S, σ_T, and $\vec{\gamma}$ (i.e. $\{\gamma_t : t = 1,\ldots,F-1\}$), as well as the AR coefficients $\vec{\alpha}$ (i.e. $\{\alpha_j : j = 1,\ldots,W_T\}$). To do this, we embed our model into a MAP framework. We assume that the prior of the features and the prior of the noise variances are uniform. Replacing (4) and (5) in (3), we formulate the MAP problem as a nonlinear and non-convex minimization problem.

$$\min_{\{\mathbb{T}_t\}_{t=1}^{F},\{\mathbb{A}_t\}_{t=1}^{F-1},\sigma_S,\sigma_T,\vec{\gamma},\vec{\alpha}} (-\ln \mathscr{L} - \ln \mathscr{P}_{\mathbb{A}} - \ln \mathscr{P}_{\mathbb{T}}) \tag{6}$$

Due to the complex form of (6), we learn the spatial layout of the DS and its dynamics in an alternating fashion. At each iteration, we either fix the dynamics and update the spatial layout or vice versa. In what follows, we show the steps involved in updating the spatial layout and the dynamics at the qth iteration. The estimate of each parameter at the qth iteration is indexed with a $[q]$ suffix.

2.4.1. Spatial layout update [qth iteration of (6)]

We employ a method similar to the one used for video object segmentation in [16] to update $\{\mathbb{T}_t[q-1]\}_{t=1}^{F}$. We will only highlight the main aspects of this method and how it applies to modeling DS's. We create a graph whose nodes are all candidates for frame-to-frame correspondences between $\{\mathbb{T}_t[q-1]\}_{t=1}^{F}$ and individual segments of these frames. In other words, a segment or swarm element in frame t corresponds to a segment or swarm element in the next frame, if the projection of the former into frame $t+1$ (according to its optical flow) overlaps with the latter. This graph allows for the clustering of similar and neighboring nodes, thus, enabling many-to-many correspondences between consecutive frames. Once this graph is created, the attributes of each node and the edge weights between neighboring nodes are determined by $s_1(.)$ and $s_2(.)$, as defined in Section 2.4. For segments that do not belong to $\{\mathbb{T}_t[q-1]\}_{t=1}^{F}$, we use identity for their transformation. Given this weighted undirected graph, we cluster its nodes into valid and invalid correspondences. This binary clustering is done using graph cuts, instead of relaxation labeling. Then, the resulting valid correspondences are broken down into individual connected components, where connectedness is over time and space. This yields $\{\mathbb{T}_t[q]\}_{t=1}^{F}$. As pointed out in [16], this method tends to cluster adjacent/occluding swarm elements with similar dynamics. For initialization, we set $\{\mathbb{T}_t[0]\}_{t=1}^{F}$ to all segments in the sequence with non-zero optical flow.

2.4.2. Dynamics update [qth iteration of (6)]

Given $\{\mathbb{T}_t[j]\}_{t=1}^{F}$, we solve (6) iteratively using Iterated Conditional Modes (ICMs) [19], which guarantees a local minimum. ICM minimizes (6) through a sequence of alternating optimization steps, where a set of variables is updated while all others remain fixed. This yields a sequence of update rules that we show next. Here, we note that we refrained from using a complete EM formulation because it adds considerable computational expense with limited performance improvement. For better readability, we will omit in what follows the $[q]$ suffix from all the parameters being estimated in the qth iteration of (6). Furthermore, we index the model parameters with $[k]$ to denote their estimates in the kth ICM iteration.

2.4.2.1 Updating variances. In the kth ICM iteration, the variances are updated to their ML estimates as follows.

$$\sigma_S^2[k] = \frac{1}{C_S} \sum_{\mathbf{N}_t^{(i)} \in \mathscr{C}} \left(\sum_{\{(tj),(tj')\} \in \Gamma_S(t,i)} w_t(j,j') \left\| \mathbf{A}_t^{(j)}[k] - \mathbf{A}_t^{(j')}[k] \right\|_F^2 \right) \tag{7}$$

$$\sigma_T^2[k] = \frac{1}{C_T} \sum_{\mathbf{N}_t^{(i)} \in \mathscr{C}} \left\| \mathbf{A}_t^{(i)}[k] - \sum_{j=1}^{\rho_t} \alpha_j \mathbf{A}_{t-j}^{(i)}[k] \right\|_F^2 \tag{8}$$

$$\gamma_t^2[k] = \frac{2}{dK_t} \sum_{i=1}^{K_t} \left\| \vec{\mathbf{f}}_{t+1}^{(i)} - \left(\mathbf{A}_t^{(i)}[k] \right) \vec{\mathbf{f}}_t^{(i)} \right\|_2^2 \tag{9}$$

2.4.2.2. Updating. $\vec{\alpha}$ We update the AR coefficients $\vec{\alpha}$ by taking the gradient of (6) with respect to $\vec{\alpha}$ and setting it to zero. This yields the linear system of equations in (10), where $\vec{\mathbf{a}}_t^{(i)} = \mathrm{vec}\left(\mathbf{A}_t^{(i)}\right)$ is the vectorized version of $\mathbf{A}_t^{(i)}$.

$$\left(\sum_{\mathbf{N}_t^{(i)} \in \mathscr{C}} \left(\mathbf{B}_t^{(i)} \right)^T \mathbf{B}_t^{(i)} \right) \vec{\alpha} = \sum_{\mathbf{N}_t^{(i)} \in \mathscr{C}} \left(\mathbf{B}_t^{(i)} \right)^T \vec{\mathbf{a}}_t^{(i)}, \text{ such that}$$
$$\mathbf{B}_t^{(i)} = \left[\vec{\mathbf{a}}_{t-1}^{(i)} | \cdots | \vec{\mathbf{a}}_{t-\rho_t}^{(i)} \right] \tag{10}$$

2.4.2.3. Updating transformations. Updating each $\mathbf{A}_t^{(i)}[k]$ requires the minimization of a convex quadratic, matrix problem. At the kth ICM iteration, we fix all of them except for $\mathbf{X} = \mathbf{A}_t^{(i)}[k]$. Here, we isolate the dependence of (6) on $\mathbf{X}$ and minimize the convex-quadratic matrix problem in (11).

$$\min_{\mathbf{X}} g(\mathbf{X}) = \frac{e_R(\mathbf{X})}{\gamma_t^2[k]} + \frac{2e_S(\mathbf{X})}{\sigma_S^2[k]} + \frac{e_T(\mathbf{X})}{\sigma_T^2[k]} \tag{11}$$

where $e_R(.)$ and $e_S(.)$ represent the reconstruction and spatial stationarity residuals, respectively. $e_T(.)$ represents the temporal stationarity residual corresponding to the frames preceding frame t. Clearly, optimal swarm element dynamics depends on a data-driven tradeoff between frame-to-frame reconstruction of features, spatial stationarity, and temporal stationarity. Minimizing $g(\mathbf{X})$ is a convex quadratic matrix program that admits a global minimum $\mathbf{X}^*$. It can be obtained using gradient descent where the rate of descent $\eta[k]$ is determined by a line search. A closed form solution for η can be derived. Till now, $\mathbf{X}$ has been an unconstrained linear transformation; however, certain applications require that it belong to a feasible set $\mathbb{S}_d$ (e.g. rotation or symmetric matrices). To do this, we project the intermediate solution at each descent step unto $\mathbb{S}_d$. In some cases, this projection is trivial. For example, if $\mathbb{S}_d = \left\{ \mathbf{X} \in \mathbb{R}^{d \times d} : \mathbf{X} = \mathbf{X}^T \right\}$ (i.e. space of symmetric matrices), the projection of X is $\mathbf{P}_{\mathbb{S}_d}(\mathbf{X}) = \frac{1}{2}\left(\mathbf{X} + \mathbf{X}^T\right)$. Using differential matrix identities, we can express the gradient of $g(\mathbf{X})$ in a computationally efficient form: $\nabla g = \left(c\mathbf{I}_d + \vec{\mathbf{b}}\vec{\mathbf{b}}^T \right)\mathbf{X} - \mathbf{D}$ where $\beta, \vec{\mathbf{b}}$, and $\mathbf{D}$ are functions of $\vec{\mathbf{f}}_t^{(i)}, \vec{\mathbf{f}}_{t+1}^{(i)}$, and the current estimates of the transformations and $\vec{\alpha}$. We can initialize $\mathbf{X}$ in the following two ways.

1. Set $\mathbf{X}_{(0)}$ equal to the transformation obtained from the previous ICM iteration (i.e. $\mathbf{X}_{(0)} = \mathbf{A}_t^{(i)}[k-1]$).
2. If $\mathbf{X}$ is constrained to be in $\mathbb{S}_d$, we set $\mathbf{X}_{(0)}$ by projecting the solution to the unconstrained version of (11), denoted as $\mathbf{X}_{\mathrm{UNC}}^*$, unto $\mathbb{S}_d$. Setting $\nabla g = \mathbf{0}_d$ and using the matrix inversion lemma, we get $\mathbf{X}_{\mathrm{UNC}}^* = \frac{1}{c}\left(\mathbf{I}_d - \frac{\vec{\mathbf{b}}\vec{\mathbf{b}}^T}{c + \|\vec{\mathbf{b}}\|_2^2} \right)\mathbf{D}$.

In our experiments, both initialization schemes have similar rates of convergence; however, (1) tends to be more numerically stable than (2) when β is small. For the first ICM iteration ($k = 0$), we initialize every $\mathbf{A}_t^{(i)}[0] = 0_d$. Numerically, we avoid division by zero by setting $\sigma_S[0] = \sigma_T[0] = \gamma_t[0] = 1$. We summarize the optimization steps of (11) in Algorithm 1. For more details, we refer the reader to A.

Algorithm 1. Gradient Descent (**GD**)

Input: $\mathbf{X}_{(0)} \in \mathbb{S}_d, \beta, \vec{\mathbf{b}}, \mathbf{D}, \epsilon$

Initialization: $\delta \leftarrow \infty; \ell = 0$
while $\delta \geqslant \epsilon$ **do**
  $\eta_\ell = \arg\min_{\eta \geqslant 0} g\left(\mathbf{X}_{(\ell)} - \eta(\nabla g)|_{\mathbf{X}_{(\ell)}}\right)$
  $\mathbf{X}_{\left(\ell+\frac{1}{2}\right)} = \mathbf{X}_{(\ell)} - \eta_l(\nabla g)|_{\mathbf{X}_{(\ell)}}$
  $\mathbf{X}_{(\ell+1)} = \mathbf{P}_{\mathbb{S}_d}\left[\mathbf{X}_{\left(\ell+\frac{1}{2}\right)}\right]$ **(optional)**
  $\delta = \frac{\|\mathbf{X}_{(\ell+1)} - \mathbf{X}_{(\ell)}\|_F}{\|\mathbf{X}_{(\ell)}\|_F}; \ell = \ell + 1$
end

Overall Learning Algorithm Algorithm 2 combines all the above update equations together to solve (6), which learns the swarm spatial layout and dynamics jointly. The worst case complexity of this algorithm is $\mathcal{O}(Fd^3)$, since it is defined by the complexity of Algorithm 1 that has a linear convergence rate.

Algorithm 2. Learn Swarm Layout and Dynamics

Input: $\{\mathbb{F}_t, \mathbb{T}_t[0], \mathbb{A}_t[0]\}_{t=1}^F, W_T, \epsilon, j_{\max}, k_{\max}$
for $j \leftarrow 0$ **TO** $j_{\max}$ **do**
  // *update spatial layout*
  • get $\{\mathbb{T}_t[j+1]\}_{t=1}^F$ from $\{\mathbb{T}_t[j], \mathbb{A}_t[j]\}_{t=1}^F$
for $t \leftarrow 1$ **TO** $F; i \leftarrow 1$ **TO** K_t **do**
    • find generalized Voronoi regions of $\mathbf{T}_t^{(i)}$
    • compute $w_t(t,i)$
end
// *update noise variances and transformations*
Initialization: $\delta \leftarrow \infty; k = 0$
while $(\delta \geqslant \epsilon)$ **AND** $(k \leqslant k_{\max})$**do**
    • compute $\sigma_S[k], \sigma_T[k], \vec{\gamma}[k], \vec{\alpha}[k]$
    for $t \leftarrow 1$ **TO** $F; i \leftarrow 1$ **TO** K_t **do**
    • compute $\beta, \vec{\mathbf{b}}, \mathbf{D}, \mathbf{X}_{(0)}$
  •$\mathbf{B}_t^{(i)}[k+1] = \mathbf{GD}\left(\mathbf{X}_{(0)}, \beta, \vec{\mathbf{b}}, \mathbf{D}, \epsilon\right)$
  end
  $\delta = \max_{(t,i)} \frac{\|\mathbf{A}_t^{(i)}[k+1] - \mathbf{A}_t^{(i)}[k]\|_F}{\|\mathbf{A}_t^{(i)}[k]\|_F}; k = k + 1$
  •$\mathbf{A}_t^{(i)}[j] = \mathbf{B}_t^{(i)}[k+1] \forall t, i$
  end
end

3. Experimental results

To validate our model and evaluate the performance of our algorithm, we conducted experiments on synthetic sequences (Section 3.1) and real sequences (Section 3.2). The synthetic sequences help provide quantitative evaluation. Our experiments show that our DS model can learn the dynamics of swarms, discriminate between different types of swarm motion, and exploit the learned dynamics/transformations to perform human action recognition.

3.1. Synthetic sequences

We construct a synthetic DS sequence of $F = 25$ frames and $K = 8$ elements (4 leaves and 4 squares with a simple textured interior). Fig. 4a shows a sample frame of this sequence, where the boundaries of the generalized Voronoi regions are drawn in green. The motion of the swarm elements is synthesized by applying a globally similar rotation $\mathscr{R}_{\theta_t^{(i)}}$. Specifically, for each element in every frame, $\theta_t^{(i)}$ is sampled from a Gaussian distribution $\mathcal{N}(\theta_0 = \frac{\pi}{25}, \sigma_\theta = \frac{1}{50})$. Noise is added to evaluate the robustness of our model and learning algorithm.

3.1.1. Model learning

The features $\vec{\mathbf{f}}_t^{(i)}$ we used were based on a polar coordinate system centered at the centroid of each element, where each angular bin had a width of $\frac{\pi}{20}$ rad. For each angular bin, we extracted two shape features (kurtosis and skew), the mean centroidal distance of the element boundary, and the mean intensity value. This yields a feature vector of size $d = 160$ that was further reduced to $d = 40$ using PCA. Setting $\epsilon = 10^{-3}, k_{\max} = 50$ and $W_T = 3$, we applied Algorithm 2 to learn the swarm dynamics. Running MATLAB on a 2.4 GHz PC, our algorithm converged in 40 ICM iterations ($\sim$10 s). Fig. 4b shows a sample transformation matrix after convergence. We evaluate our model fitting performance by using three measures: the reconstruction residual error $\zeta_R(t)$ defined in (12), the spatial residual error $\zeta_S(t)$ defined in (13), and the temporal residual error $\zeta_T(t)$ defined in (14).

$$\zeta_R(t) = \frac{1}{K}\sum_{i=1}^{K} \frac{1}{\|\vec{\mathbf{f}}_t^{(i)}\|_2} \sqrt{e_R\left(\mathbf{A}_t^{(i)}\right)} \tag{12}$$

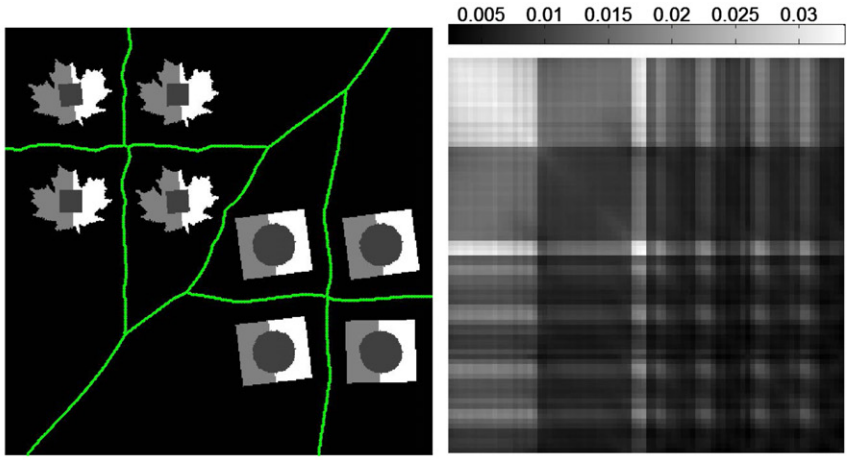
$$\zeta_S(t) = \frac{1}{K}\sum_{i=1}^{K} \frac{1}{|\Gamma_S(t,i)| \|\mathbf{A}_t^{(i)}\|_F} \sqrt{e_S\left(\mathbf{A}_t^{(i)}\right)} \tag{13}$$

$$\zeta_T(t) = \frac{1}{K}\sum_{i=1}^{K} \frac{1}{|\Gamma_T(t,i)| \|\mathbf{A}_t^{(i)}\|_F} \sqrt{e_T\left(\mathbf{A}_t^{(i)}\right)} \tag{14}$$

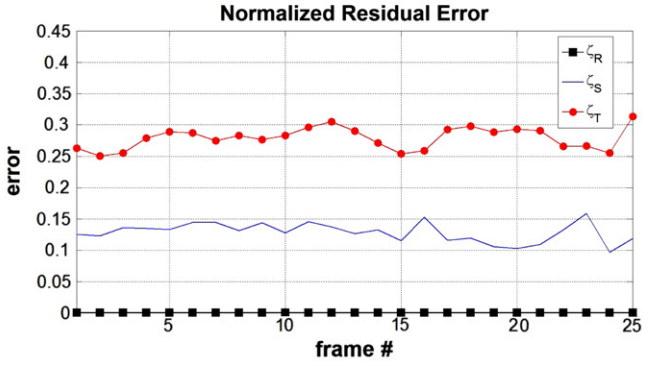
These residuals quantify the average normalized error incurred in reconstructing the data and enforcing stationarity in the spatiotemporal neighborhood of each swarm element. Clearly, the smaller these measures are, the better our model fits the data. Fig. 4c plots these measures for all frames in the sequence. All three measures show a stable variation with time. ζ_S and ζ_T are consistently larger than ζ_R due to the added noise corrupting each transformation. In fact, as $\sigma \to 0, \zeta_S$ and ζ_T both get closer to ζ_R. Furthermore, ζ_T is consistently larger than ζ_S because temporal neighborhoods only extend $W_T = 3$ frames from each swarm element. In fact, as $W_T \to (F-1), \zeta_T$ gets closer to ζ_S, since temporal stationarity is enforced on a larger number of frames. Here, we point out that although the leaf and square elements are significantly different in appearance, their dynamics are the same. This reinforces the fact that our method successfully separates between swarm appearance and dynamics. In Fig. 4d, we plot the average residuals as a function of increasing noise level. The error residuals increase with the noise level in a super-linear fashion, thus, providing empirical evidence that our fitting algorithm is relatively stable under transformational noise.

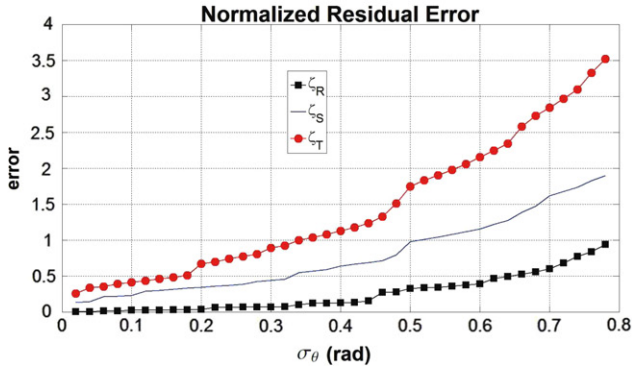
3.1.2. Motion discrimination

Here, we demonstrate that the learned transformations can discriminate between different types of motion. Another synthetic DS sequence is constructed in the same manner as before, but with the leaf and square elements now rotating in opposite directions. Leaf elements undergo $\mathscr{R}_{\theta_t^{(i)}}$, while square elements undergo $\mathscr{R}_{-\theta_t^{(i)}}$. After learning the swarm dynamics, we compute all the distances (i.e. Frobenius norm of the difference) between pairs of learned transformations. We show the resulting distance matrix in Fig. 5a. We see that the transformations corresponding to the leaf elements are close to each other and far from those corresponding to the square elements. For visualization purposes, we perform MDS on these pairwise distances to embed the transformations in $\mathbb{R}^3$. In this space, the leaf and square dynamics are easily separable. Moreover, these transformations can be perfectly clustered using spectral clustering ($K = 2$) [20]. This result reinforces the fact that our method can successfully learn and dis-



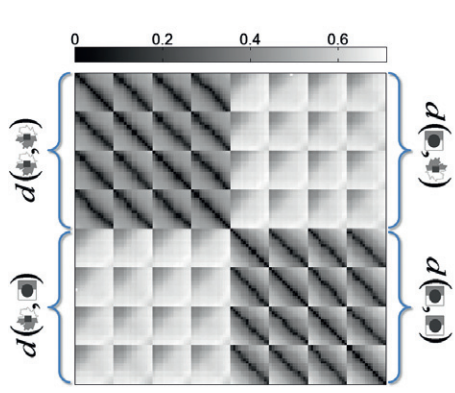
(a) sample frame (b) learned transformation

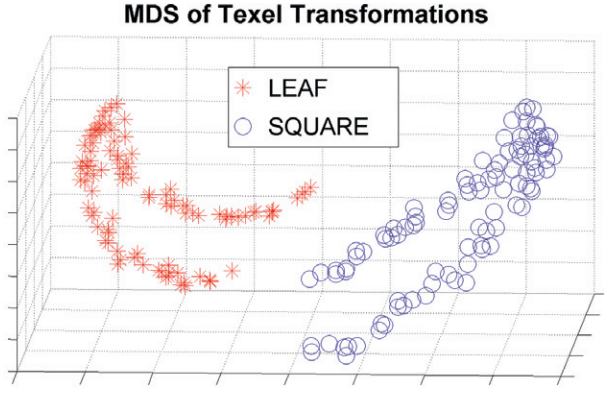




(c) modeling performance for $\sigma_\theta = \frac{1}{50}$ noise level (d) modeling performance for multiple σ levels

Fig. 4. (a) Is a frame in the synthetic sequence, (b) shows transformation $\mathbf{A}_{10}^{(1)}$, after convergence, (c) plots the spatiotemporal residual errors for all frames at a single noise level σ, while (d) plots the average residuals with increasing noise. All video results are provided in Supplementary material.





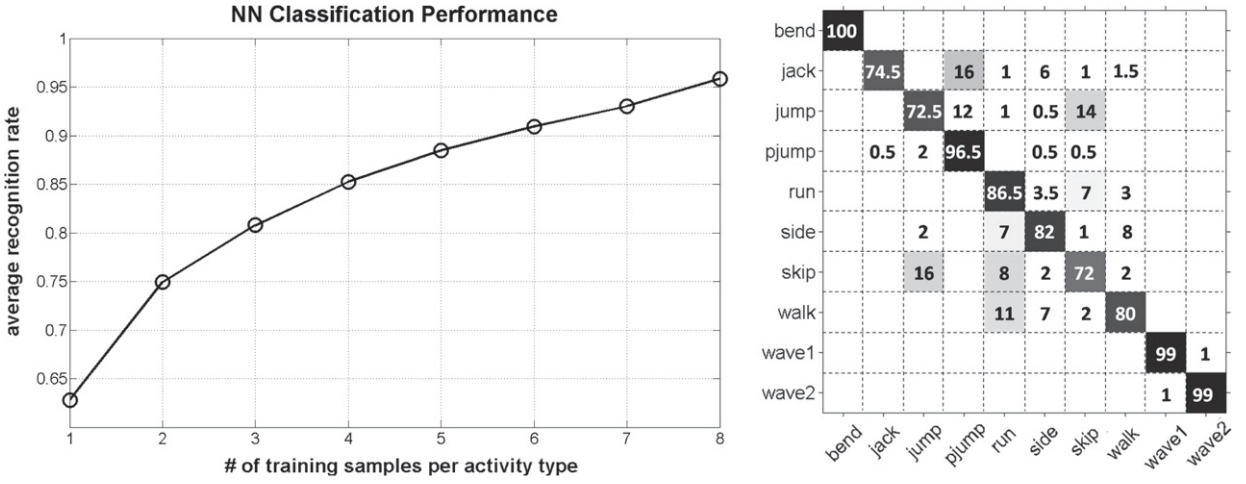
(a) distance matrix (b) MDS of swarm dynamics

Fig. 5. (a) Shows the distances between the swarm dynamics in the synthetic sequence. Note that brighter values designate larger distances. (b) Projects the transformations onto $\mathbb{R}^3$ using MDS.

criminate between different motions occurring within a single DS sequence. This conclusion is valid as long as the "neighborness" weights associated with swarm elements undergoing similar dynamics are reasonably higher than those undergoing different dynamics.

3.2. Real sequences

In this section, we present experimental results produced when Algorithm 2 is applied to real sequences where single or multiple elements are undergoing an underlying dynamic swarm motion.



(a) NN Recognition Performance

(b) Confusion matrix using 4 training samples per activity type

Fig. 6. (a) Plots the recognition performance of a NN classifier vs. the number of training samples used per action type. (b) Shows the confusion matrix when eight samples are used for training on the Weizmann action recognition dataset. Darker squares indicate higher percentages.

Fig. 7. "Birds", "geese", "robot", and "pedestrian" swarms.

3.2.1. Single swarm element sequences

Here, we apply our algorithm to human action recognition, where we consider the human as a single texel. There is no need to determine the spatial neighborhoods of the texels. The action sequences were obtained from the Weizmann classification database,[3] which contains 10 human actions. We use background subtraction to extract the texels. In addition to the features used earlier, we use the height and the width of the texel masks at each frame.

After learning the texel transformations, we use a nearest neighbor (NN) classifier to recognize a test action sequence, given a set of training sequences. We define the dissimilarity between two sequences ($\mathscr{S}_1$ and $\mathscr{S}_2$) as the dynamic time warping (DTW) cost [21] needed to warp the transformations of $\mathscr{S}_1$ into those of $\mathscr{S}_2$, where the dissimilarity between transformations $\mathbf{X}_1$ and $\mathbf{X}_2$ is defined as: $d(\mathbf{X}_1,\mathbf{X}_2) = 1 - \frac{\text{trace}(\mathbf{X}_1^T\mathbf{X}_2)}{\|\mathbf{X}_1\|_F\|\mathbf{X}_2\|_F}$. Such a warping is crucial, since $\mathscr{S}_1$ and $\mathscr{S}_2$ might have different cardinalities (i.e. swarm elements do not have to appear in the same number of frames). The DTW cost is efficiently computed using dynamic programming. Fig. 6a plots the variation of the average recognition rate vs. the number of sequences (per action class) used for training. For each training sample size, we randomly choose a set of such size from each action class and perform classification. We repeat this multiple times and average the recognition rate to obtain the plotted values. Obviously, the performance improves as the number of training samples increases. More importantly, we note that a simple classifier using only one training sample achieves a 62% recognition rate, where random chance is 10%. Furthermore, Fig. 6b shows the average confusion matrix. Note the high diagonal values. Here, we point out that confusion occurred between similar actions especially for the ("jump", "skip") and ("run", "walk") pairs. Although the performance of our DS method on this dataset is not better than state-of-the-art approaches that make use of specialized features, improved performance is expected, when texels are extracted more reliably and features are more discriminative of human appearance/motion.

Table 1
Average normalized residual error (as %). The percentage values in parentheses are the average errors normalized by the error incurred when the swarm dynamics are not updated.

	"Birds"	"Geese"	"Robot"	"Pedestrian"
ζ_R	8.2	10.3	3.5	12.9
ζ_S	12.5	6.5	11.6	15.8
ζ_T	18.0	14.1	16.4	23.1
$\frac{\zeta_R}{\zeta_R(\mathbf{I}_d)}$	5.4%	4.9%	4.2%	9.5%
$\frac{\zeta_S}{\zeta_S(\mathbf{I}_d)}$	6.8%	5.8%	5.5%	11.6%
$\frac{\zeta_T}{\zeta_T(\mathbf{I}_d)}$	4.1%	7.7%	4.4%	18.3%

3.2.2. Multiple swarm element sequences

We apply our algorithm to swarm video sequences compiled from online sources. We perform model learning and motion discrimination on four sequences: "birds" [14], "geese", "robot swarm",[4] and "pedestrian" [5].

[3] These sequences are publicly available at www.wisdom.weizmann.ac.il/vision/SpaceTimeActions.html.

[4] These sequences are publicly available at http://people.csail.mit.edu/jamesm/swarm.php/videos.

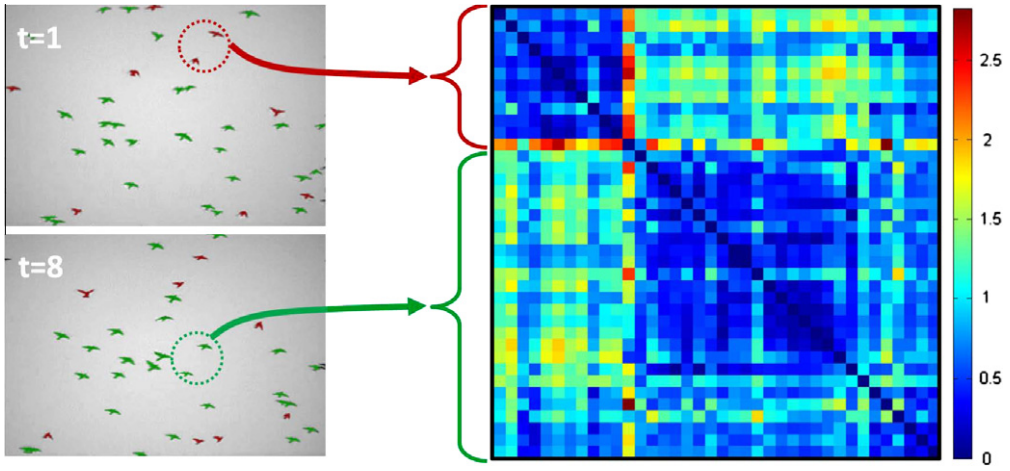


Fig. 8. Shows the "birds" swarm example containing a "bird-flapping" and "bird-gliding" motion. The pairwise distances between the learned transformations are shown on the right.

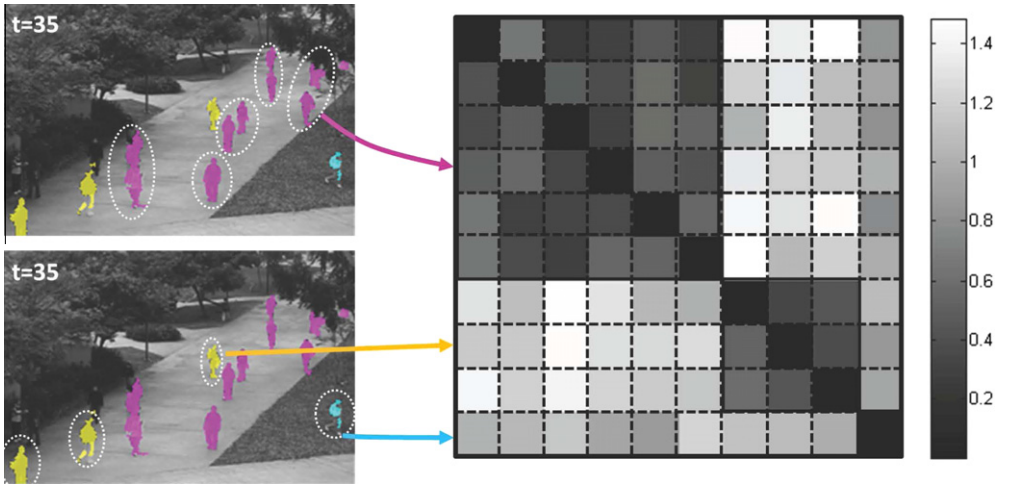


Fig. 9. Shows a pedestrian example containing three types of motion. The extracted swarm elements are color-coded. The pairwise distances between the learned transformations are shown on the right. Brighter squares indicate larger distances. Refer to the Supplementary material for these and other video results.

3.2.2.1. Model learning. Here, the features we used were the same as in Section 3.2.1. Setting $\epsilon = 10^{-3}, j_{\max} = 5, k_{\max} = 50$ and $W_T = 5$, we applied Algorithm 2 to learn the spatial layout and dynamics of each swarm sequence. To evaluate the performance of our method, we conducted a leave-*N*-out experiment, where we learn the swarm dynamics using all the frames except for *N*. In this experiment, we set $N = \frac{F}{W_T}$. The transformations and features of the elements in the left out frames are reconstructed using the AR model. We repeated this experiment and reported the average normalized residual errors in Table 1, for the four sequences. These results show that our DS model represents the ground truth data well. Here, we note that the error was the highest for the "pedestrian" sequence due to the variability in the swarm dynamics and appearance. Also, we compared these residual errors to the case when identity is used instead of the learned transformations (i.e. no dynamics update). The ratio of these two errors are shown in parenthesis (in %). Clearly, our learned dynamics enable reliable model fitting (see Fig. 7).

3.2.3Motion discrimination

Here, we demonstrate that our method can discriminate between different motions (i.e. sequences of transformations) within the same video sequence. After learning the swarm dynamics, we compute the dissimilarity in dynamics between every pair of swarm elements. We use DTW to compute the dissimilarity between two sequences of swarm element transformations and use the DTW costs in a spectral clustering setting to cluster the swarm elements' dynamics, as in Section 3.2.1.

The "birds" and "pedestrian" sequences contain more than one distinguishable motion. Fig. 8 illustrates the clustering results obtained for the "birds" sequence. The extracted swarm elements are color-coded in the frames according to their distinct motions. In this sequence, two types of motion co-exist: **(i)** a "bird-flapping" motion where wings oscillate up and down and **(ii)** a "bird-gliding" motion where the wings remain relatively still. On the right, Fig. 8 shows the DTW distances computed between all pairs of swarm element dynamics. We clearly see that type **(i)** elements undergo quite different transformations than those of type **(ii)**. Our approach was able to simultaneously learn the different dynamics in the sequence and discriminate them. This cannot be done by DT models such as [14].

We also apply our algorithm to "pedestrian" video sequences, where humans or groups of humans are considered swarm elements. These sequences were obtained from the UCSD pedestrian traffic database [5]. Fig. 9 illustrates the results obtained for a single pedestrian sequence that exhibits dense swarm activity.

The extracted swarm elements are color-coded in the frames according to their distinctive dynamics. In this sequence, three types of motion co-exist. **(i)** Elements (some of which are groups of pedestrians) move/walk from the top right corner to the bottom left corner. **(ii)** Other elements moves in the opposite direction. **(iii)** One element represents a person crossing the grass instead of walking along the diagonal path. On the right, Fig. 9 shows the DTW distances computed between all pairs of swarm elements. We see that the elements of **(i)** undergo much more similar transformations than those of **(ii)–(iii)**, which, in turn, have significantly different dynamics. Some pedestrian segments were not part of the spatial layout since they were indistinguishable from the background.

4. Conclusion

This paper proposes a spatiotemporal model for learning the spatial layout and dynamics of elements in swarm sequences. It represents a swarm element's motion as a sequence of linear transformations that reproduce its properties subject to local stationarity constraints. We conducted experiments on real sequences to demonstrate our approach's merit in representing swarm dynamics and discriminating between different dynamics. Our future goal is to apply this method to motion synthesis and recognition.

Acknowledgment

The support of the Office of Naval Research under Grant N00014-09-1-0017 and the National Science Foundation under grant IIS 08-12188 is gratefully acknowledged.

Appendix A. Optimization details of (11)

The cost function $g(\mathbf{X})$ in (11) is convex quadratic in $\mathbf{X} = \mathbf{A}_t^{(i)}[k]$,[5] since it is a non-negative linear combination of three error terms (e_R, e_S, and e_T), which are convex quadratic in their own right. We express these terms and their gradients with respect to $\mathbf{X}$ as follows.

$$e_R(\mathbf{X}) = \left\| \vec{\mathbf{f}}_{t+1}^{(i)} - \mathbf{X}\vec{\mathbf{f}}_t^{(i)} \right\|_2^2 \tag{A.1}$$

$$e_S(\mathbf{X}) = \sum_{(t,i')\in\Gamma_S(t,i)} w_t(i,i') \left\| \mathbf{X} - \mathbf{A}_t^{(i')}[k] \right\|_F^2 \tag{A.2}$$

$$e_T(\mathbf{X}) = \|\mathbf{X} - \Sigma_1[k]\|_F^2 + \sum_{j=1}^{\min(W_T, F-t-1)} \left\| \alpha_j \mathbf{X} - \Sigma_2^{(j)}[k] \right\|_F^2, \text{ where } \begin{cases} \Sigma_1[k] = \sum_{j=1}^{\rho_t} \alpha_j \mathbf{A}_{t-j}^{(i)}[k] \\ \Sigma_2^{(j)}[k] = \mathbf{A}_{t+j}^{(i)}[k] - \sum_{m=1; m\neq j}^{\rho_{t+j}} \alpha_m \mathbf{A}_{t+j-m}^{(i)}[k] \end{cases} \tag{A.3}$$

Next, we find an explicit expression for the gradient of $g(\mathbf{X})$ using the gradients of the three residual terms above.

$$\nabla e_R = -2\vec{\mathbf{f}}_t^{(i)}\vec{\mathbf{f}}_{t+1}^{(i)T} + 2\left(\vec{\mathbf{f}}_t^{(i)}\vec{\mathbf{f}}_t^{(i)^T}\right)\mathbf{X} \tag{A.4}$$

$$\nabla e_S = -2 \sum_{(t,i')\in\Gamma_S(t,i)} w_t(i,i')\mathbf{A}_t^{(i')}[k] + 2\left(\sum_{(t,i')\in\Gamma_S(t,i)} w_t(i,i') \right)\mathbf{X} \tag{A.5}$$

$$\nabla e_T = -2\left(\Sigma_1[k] + \sum_{j=1}^{\min(W_T, F-t-1)} \alpha_j \Sigma_2^{(j)}[k] \right) + 2\left(1 + \sum_{j=1}^{\min(W_T, F-t-1)} \alpha_j^2 \right)\mathbf{X} \tag{A.6}$$

$$\therefore \nabla g = \frac{\nabla e_R}{\gamma_t^2[k]} + \frac{2\nabla e_S}{\sigma_S^2[k]} + \frac{\nabla e_T}{\sigma_T^2[k]} = \left(c\mathbf{I}_d + \vec{\mathbf{b}}\vec{\mathbf{b}}^T \right)\mathbf{X} - \mathbf{D} \tag{A.7}$$

where

$$\vec{\mathbf{b}} = \frac{\vec{\mathbf{f}}_t^{(i)}}{\gamma_t[k]}; c = \frac{\sum_{(t,i')\in\Gamma_S(t,i)} w_t(i,i')}{\sigma_S^2[k]} + \frac{1 + \sum_{j=1}^{\min(W_T, F-t-1)} \alpha_j^2}{\sigma_T^2[k]} \tag{A.8}$$

$$\mathbf{D} = \frac{\vec{\mathbf{f}}_t^{(i)}\vec{\mathbf{f}}_{t+1}^{(i)T}}{\gamma_t^2[k]} + \frac{\sum_{(t,i')\in\Gamma_S(t,i)} w_t(i,i')\mathbf{A}_t^{(i')}[k]}{\sigma_S^2[k]} + \frac{\Sigma_1[k] + \sum_{j=1}^{\min(W_T, F-t-1)} \alpha_j \Sigma_2^{(j)}[k]}{\sigma_T^2[k]} \tag{A.9}$$

Setting the gradient to zero and using the matric inversion lemma, we obtain a closed form expression of the global minimum for the unconstrained version of (11), as stated in (A.10). This solution is referred to in Section 2.4.2.

$$\mathbf{X}_{\text{UNC}}^* = \frac{1}{c}\left(\mathbf{I}_d - \frac{\vec{\mathbf{b}}\vec{\mathbf{b}}^T}{c + \|\vec{\mathbf{b}}\|_2^2} \right)\mathbf{D} \tag{A.10}$$

Appendix B. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.cviu.2012.09.002.

References

[1] N. Ahuja, S. Todorovic, Extracting texels in 2.1d natural textures, in: International Conference on Computer Vision, 2007.
[2] M. Szummer, R.W. Picard, Temporal texture modeling, in: International Conference on Image Processing, vol. 3, 1996.
[3] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, M. Werman, Texture mixing and texture movie synthesis using statistical learning, IEEE Trans. Visual. Comput. Graph. (2001) 120–135.
[4] S. Soatto, G. Doretto, Y.N. Wu, Dynamic textures, Int. J. Comput. Vis. 51 (2003) 91–109.
[5] A.B. Chan, N. Vasconcelos, Modeling, clustering, and segmenting video with mixtures of dynamic textures, IEEE Trans Pattern Anal. Mach. Intell. 30 (5) (2008) 909–926.
[6] A. Ravichandran, R. Chaudhry, R. Vidal, View-invariant dynamic texture recognition using a bag of dynamical systems, in: Conference on Computer Vision and, Pattern Recognition, 2009, pp. 1651–1657.
[7] A. Chan, N. Vasconcelos, Layered dynamic textures, in: Neural Information Processing Systems, 2006, pp. 203–210.
[8] A. Chan, N. Vasconcelos, Variational layered dynamic textures, in: Conference on Computer Vision and, Pattern Recognition, 2009, pp. 1062–1069.
[9] C.-B. Liu, R. sung Lin, N. Ahuja, Modeling dynamic textures using subspace mixtures, in: International Conference on Multimedia & Expo, 2005, pp. 1378–1381.
[10] K. Derpanis, R. Wildes, Dynamic texture recognition based on distributions of spacetime oriented structure, in: Conference on Computer Vision and, Pattern Recognition, 2010, pp. 191–198.
[11] Y. Xu, Y. Quan, H. Ling, H. Ji, Dynamic texture classification using dynamic fractal analysis, in: International Conference on Computer Vision, 2011, pp. 1219–1226.
[12] B. Ghanem, N. Ahuja, Extracting a fluid dynamic texture and the background from video, in: Conference on Computer Vision and, Pattern Recognition, 2008.
[13] Y. Wang, S.-C. Zhu, Analysis and synthesis of textured motion: particles and waves, IEEE Trans. Pattern Anal. Mach. Intell. (2004) 1348–1363.
[14] S.-C. Zhu, C. en Guo, Y. Wang, Z. Xu, What are textons?, Int J. Comput. Vis. (2005) 121–143.
[15] M. Yang, T. Yu, Y. Wu, Game-theoretic multiple target tracking, in: International Conference on Computer Vision, 2007.
[16] W. Brendel, S. Todorovic, Video object segmentation by tracking regions, in: International Conference on Computer Vision, 2009.
[17] E. Akbas, N. Ahuja, From ramp discontinuities to segmentation tree, in: Asian Conference on Computer Vision, 2009.
[18] Q. Du, V. Faber, M. Gunzburger, Centroidal voronoi tessellations: applications and algorithms, SIAM Rev. 41 (4) (1999) 637.
[19] J. Besag, On the statistical analysis of dirty pictures, J. R. Stat. Soc. 48 (3) (1986) 259–302.
[20] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, Adv. Neural Inform. Process. Syst. 17 (1601–1608) (2004) 16.
[21] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoustics Speech Signal Process. 26 (1978) 43–49.

[5] Note that this problem is being solved at the kth ICM iteration.