Unsupervised Identification of Coherent Motion in Video

Luciano Silva da Silva and Jacob Scharcanski
Universidade Federal do Rio Grande do Sul
Instituto de Informática
Caixa Postal 15064, 91501-970, Porto Alegre, RS, Brasil
{lssilva,jacobs}@inf.ufrgs.br

Abstract

The identification and classification of motion patterns in point trajectories has been an important issue in understanding and representing dynamic scenes. This paper proposes an unsupervised approach to identify coherent motion in video. Instead of producing a spatio-temporal segmentation of the raw data, the proposed method analyzes point trajectories along the video sequence to identify sets of points that move coherently. This new way of extracting motion information from videos potentially can be used in different areas of image processing and computer vision.

1. Introduction

Motion analysis and representation are some of the most challenging problems in video understanding. In recent years, many methods have been proposed to segment moving objects in video sequences through curve evolution, object tracking and level sets [6, 14, 3]. An interesting method for video-object segmentation based on level sets is presented in [6]. In this method, occlusion is handled by approaching 2D object tracking as a 3D volume problem, and the video sequence is divided into motion-objects volumes, occlusion volumes and disocclusion volumes. Active contours also have been employed to track rigid and nonrigid objects [3], and in this case the weights of the energy terms are determined adaptively with a performance metric. A limitation of the methods based on curve evolution is that they only do tracking of region boundaries along the video sequence, and illumination changes, occlusions, or complex regions easily mislead the tracking process. Besides, recovery from tracking errors is not trivial (except when object models are known a priori).

On the other hand, moving structures can be segmented based on the trajectories of their samples (i.e. points), which reduces to finding a linear manifold [11]. In [13] a factorization method is used to segment a wide range of motions (i.e. moving structures), such as independent, articulated, rigid, non-rigid, degenerate, non-degenerate, or any combination of them. However, this method requires that points are known in advance in all frames and, therefore, does not handle occlusion.

In this paper, we propose to segment moving structures in videos by tracking the trajectories of *point markers*, namely particles, linked to those structures. The main advantage of the proposed approach is that we do not need to assume a specific type of motion to detect it (although, we do need to assume that motion changes smoothly between adjacent frames). Points that are moving coherently along a common trajectory are clustered, and with these clusters we identify moving structures in the video sequence. The trajectories of points are determined using the method presented by Sand and Teller [8], in which point sampling is adaptive to the local image contents and trajectories are estimated using only information at point level (i.e. it does require patch-based features [9]). In the proposed method, video objects are tagged with long-range markers, represented by particles that follow similar trajectories, and are clustered together.

Because of its flexibility, our method potentially can handle structures with complex shapes and free-form motion. In video editing applications, the long-range correspondence of points and their coherent motion can provide ways of implementing high-level edition tasks. Although unsupervised clustering of point trajectories has already been presented in [13], to the best of our knowledge the present work is the first to approach this problem considering arbitrary particle lifetimes and occlusions. The method outlined here could be useful in several applications, such as coding moving objects in videos, or generating descriptors for the video contents.

This paper is organized as follows. The adaptive creation of particles, i.e. point sampling, and point tracking are described in Section 2. Section 3 discusses how points in coherent motion are clustered. Our preliminary experimental results are presented in Section 4. Finally, Section 5 summarizes the main results of this paper, discusses some limitations of the proposed approach and suggests future work directions.

2. Estimation of Particle Trajectories

We estimate particle trajectories with the Particle Video approach proposed by Sand and Teller [8]. In their approach, video motion is represented by a set of particles that move along the video sequence. These particles are created by interpolating image point samples, which is different from the feature patch concept [9], since the latter is based on the neighborhood of pixels.

The particle density is adaptive and depends on the image content. Regions with more content in higher frequencies tend to have more particles than the regions with more content in lower frequencies, so it is easier to model complex motion patterns. Particles are included in the set (added) whenever (i.e. in video frames) and wherever (i.e., at image points) the maximum particle proximity criterion is satisfied. After created and inserted in a frame, particles are tracked in the subsequent frames, until the video points they represent are occluded or surpass the visible frame area. In this work, we only use one pass forward over the entire video. However, more passes can be performed (forward and backward), aiming at best distribution and localization of particles, at the cost of increasing computational complexity.

The estimation of particles trajectories used in this work can be divided in four stages: particle addition, particle propagation, particle pruning and particle location optimization (the third and fourth stages differ from the stages with similar nomenclature described in [8]). Next, these four stages are described in more detail.

2.1. Particle Addition

Let $I(x, y, t)$ be a video sequence, defined over spatial coordinates x and y and the temporal coordinate t. Here, we represent one frame of the sequence simply by $I(x, y)$. To determine whether new particles must be added to a frame $I(x, y)$, we filter it with a stack of Gaussian kernels $\{\sigma(j) = 1.9^j | 0 \leq j \leq 5\}$, producing a set of images $\{I_j(x, y)\}$, where j is the index of the correspondent Gaussian kernel scale $\sigma(j)$.

For each pixel (x, y) of $I(x, y)$, we determine the range of scales $\sigma(j)$ over which the correspondent filtered intensity values $I_j(x, y)$ are approximately constant. We assume that a pixel at (x, y) is approximately constant over j scales if $||I_j(x, y) - I_1(x, y)||_2 < 10$ and j is the scale maximum (i.e., $||I_{j+1}(x, y) - I_1(x, y)||_2 \geq 10$). Let us define $z(x, y) = j$ as the index map, and $s(x, y) = \sigma(j)$ as the scale map, both information will help in the process of adding new particles to a video frame $I(x, y)$. Here and thereafter, a video frame $I(x, y)$ is defined as a 3-channel image, where the first channel is the image brightness (whose values are within the interval $[0, 255]$), the second channel is the green minus red color component, and the third is the green minus blue color component. The color difference channels are scaled by 0.25 to reduce the impact of color sampling and compression artifacts [7].

To obtain a smooth particle spatial distribution, the index map $z(x, y)$ is filtered using a Gaussian kernel with $\sigma_z = 3$ and rounded to integer values, resulting in the filtered index map $\hat{z}(x, y)$. Then, the correspondent smooth scale map $\hat{s}(x, y) = \sigma(\hat{z}(x, y))$ is obtained.

Adding particles to a frame is an iterative process. A particle is inserted in a pixel location (x, y) if its distance to the nearest particle is greater than $\hat{s}(x, y)$; new particles are inserted until no pixel location satisfies this particle minimum distance condition. In the first frame, the particle addition process starts with no particles. In subsequent frames, the process starts with the set of particles that *survived* (i.e., were not occluded or left the visible area) from the last frame. This point sampling technique is similar to that used in [8].

2.2. Particle Propagation

In the particle propagation stage, the displacement of particles from the current to the subsequent frame - in the forward video sequence - are estimated. To propagate a particle n from frame t to frame $t + 1$, we use the flow field specified by the horizontal component displacement field $\bar{u}(x, y, t)$, and the vertical component displacement field $\bar{v}(x, y, t)$:

$$x_n(t+1) = x_n(t) + \bar{u}(x_n(t), y_n(t), t),$$
$$y_n(t+1) = y_n(t) + \bar{v}(x_n(t), y_n(t), t).$$

Any optical flow technique can be used to obtain the displacement fields $\bar{u}$ and $\bar{v}$ at the pixel locations. At the particle locations, the flow values are obtained by bilinear interpolation. In this work, we use the variational optical flow method proposed by Sand and Teller. Implementation issues can be found in [8] and [7].

2.3. Particle Pruning

Unlike the approach proposed by Sand and Teller, we only prune particles when they become occluded or leave the field of view. We define the field of view as being the set of points that are at least 5 pixels distant from the image borders. So, when a particle leaves this field of view after the optical flow propagating stage, it is removed from the current frame. This avoids propagating incorrect optical flow estimates in regions of higher uncertainty.

To detect when particles become occluded, we use an occlusion labeling strategy employed by the variational optical flow calculation [8]. In order to determine the regions of the current frame regions that will become occluded in the subsequent frame, the flow field divergence and pixel projection error are combined as detailed next. Let us define the optical flow divergence as:

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

Since we are only interested in the regions that will become occluded, the positive divergence values are discarded:

$$r'_d(x,y,t) = \begin{cases} r_d(x,y,t), & r_d(x,y,t) < 0 \\ 0, & \text{otherwise} \end{cases}$$

Let us define the pixel projection error as:

$$r_e(x,y,t) = \\ ||I(x,y,t) - I(x+\bar{u}(x,y,t), y+\bar{v}(x,y,t), t+1)||_2.$$

The optical flow divergence is combined with the pixel projection error as proposed in [8], so we can predict which particles will become occluded in subsequent frames (i.e. based on the obtained motion compensation error, and the information that regions tagged by these particles are moving in opposite directions, leading to an occlusion in subsequent frames):

$$r(x,y,t) = e^{-\left(\frac{|r'_d(x,y,t)|}{2\cdot\sigma_d^2} + \frac{r_e(x,y,t)}{2\cdot\sigma_e^2}\right)}.$$

The nearer to '0' is $r(x,y,t)$, the stronger is the likelihood that a location (x,y) will be occluded in frame $t+1$. In all our experiments we chose $\sigma_d = 0.3$, $\sigma_e = 20$, and a particle n in frame $t+1$ is eliminated if $r(x_n, y_n, t) < 0.5$ (i.e. became occluded).

In this work, if a particle becomes occluded or leave the visible field, we eliminate the particle and it is no longer considered in the tracking process, even if the image location it represents becomes disoccluded later. In some video understanding tasks, however, it is important to track moving objects that are becoming occluded, and usually an object model is known a priori. The algorithm can be modified to handle these situations, keeping track of hidden particles where they satisfy the object model constraints.

2.4. Particle Location Optimization

In the particle location optimization stage, the position of each particle n in the current frame t is fine tuned by minimizing the objective function:

$$E(n,t) = \sum_{m=1}^{3}(E_d^{[m]}(n,t)) + \alpha_f E_f(n,t), \qquad (1)$$

where m denotes the image channels, E_d and E_f are the data-matching and the flow-matching terms, respectively; α_f provides a tradeoff between the two terms and is set to 0.5.

The data-matching term measures the projection error of a particle n in frame t for each image channel m, and is defined as:

$$E_d^{[m]}(n,t) = \\ \Psi([I^{[m]}(x_n(t), y_n(t), t) - I^{[m]}(x_n(t_n^0), y_n(t_n^0), t_n^0)]^2), \qquad (2)$$

where t_n^0 is the first frame of the sequence in which the particle n appears, and $\Psi(\beta^2) = \sqrt{\beta^2 + \epsilon^2}$, with $\epsilon = 0.001$, is a robust norm [1] in which outliers have less influence than with the standard L^2 norm.

The flow-matching term acts like a positional constraining term, avoiding particle to be displaced far away from the position predicted by the optical flow vectors. This term is defined as:

$$E_f(n,t) = \\ \Psi([\bar{u}(x_n(t-1), y_n(t-1), t-1) - u_n(t)]^2 \\ + [\bar{v}(x_n(t-1), y_n(t-1), t-1) - v_n(t)]^2), \qquad (3)$$

where $u_n(t) = x_n(t) - x_n(t-1)$ and $v_n(t) = y_n(t) - y_n(t-1)$.

Let $dx_n(t)$ and $dy_n(t)$ be the horizontal and the vertical displacements of a particle n, respectively. We are interested in fine-tuning the location $(x_n(t), y_n(t))$ of this particle in frame t. If we substitute $x_n(t)$ by $x_n(t) + dx_n(t)$ and $y_n(t)$ by $y_n(t) + dy_n(t)$ in Equations 2 and 3, and derive the objective function E in relation to $dx_n(t)$ and $dy_n(t)$, we obtain a system of equations that can be solved for $dx_n(t)$ and $dy_n(t)$:

$$\left\{\frac{\partial E}{\partial dx_n(t)} = 0, \frac{\partial E}{\partial dy_n(t)} = 0\right\}.$$

Since E is highly nonlinear, the minimization is not trivial. Thus, we linearize the data [1] to obtain a system of equations, with two equations for each particle (one for $dx_n(t)$ and another for $dy_n(t)$).

The system of equations can be solved for $dx_n(t)$ and $dy_n(t)$ using the fixed-point optimization method proposed in [8].

After solving the system of equations above, the particle positions can be updated as follows:

$$x_n(t) \leftarrow x_n(t) + dx_n(t), \\ y_n(t) \leftarrow y_n(t) + dy_n(t).$$

The positions of all particles in frame t are now defined. If necessary, the algorithm adds more particles to this frame (see Section 2.1), and then the process continues in the subsequent frames up to the end of the video.

3. Identification of Particles in Coherent Motion

The representation of motion in videos through particle trajectories should be flexible, allowing to identify unconstrained motion of complex structures and to occlusions, with adaptive granularity. However, extracting high-level information about the moving structures is not a trivial task for the reasons described next.

Different particles assigned to the same moving structure can have different lifetimes. For instance, a pair of particles in coherent motion (e.g. two particles assigned to the same moving structure) may never coexist in the same frame (due the their different lifetimes). Besides, the incidence of outliers and erroneous motion patterns is unavoidable (because of noise, artifacts caused by lighting and quantization/compression, and the absence of a validation model for the motion of particles).

Only particles appearing simultaneously in at least two consecutive frames can have their motion patterns compared directly. However, particles that coexist in frame t, also tend to coexist with other particles in other frames $t^{'}$. So, motion patterns can be compared continuously, at any frame t, by keeping track in each vicinity of how particles move in relation to each other in frame t, and how these motion patterns change with respect to other particles that only exist in different frames $t^{'}$. The idea is similar to the combination of data partitions in ensemble clustering [4, 10, 12].

According to the ensemble clustering philosophy, given a data set with N samples, there are different ways of partitioning this data set:

- Applying different clustering algorithms;
- Applying the same clustering algorithms with different parameters;
- Applying clustering algorithms to different data representations (for instance, using different feature spaces);
- Applying clustering algorithms to different data partitions (i.e. to subsets of the N samples.)

The last option describes the approach adopted for grouping particles that are in coherent motion. We apply a clustering algorithm to each set of particles that coexist (i.e. have lifetime intersection), and then combine these subset clusterings into larger clusters of particles.

Most ensemble clustering methods available require the computation of the pairwise similarity for all objects in the collection. In our case, the number of particles in a video can be large, and the computational cost could become unacceptable.

An interesting option is to consider the integration of different data partitions as a cluster correspondence problem [10], where groups of similar clusters are identified and combined (forming meta-clusters). Next, we present an algorithm to combine several clusters of particles into meta-clusters, using a version of the ensemble clustering method proposed in [10], where the selection of a suitable the number of meta-clusters is automatic.

Particle clustering inconsistencies are expected, because in a large set of particles moving stochastically some tracking errors usually occur. For this reason, the clustering step is followed by cluster validation, where particle context, motion and spatial location are combined to check for possible inconsistencies. Finally, a spatial filtering step is performed to eliminate outliers and groups of particles that are not considered significant.

3.1. Ensemble Clustering of Particles

In order to cluster the particles inserted in a video sequence, we first identify the subsets of particles that present similar motion patterns in adjacent frames. Let $P = p_1, p_2, ..., p_N$ be the complete set of N particles in the video, and $\vec{g_n}(t, t+l)$ denote the displacement vector of particle n, between frame t and frame $t+l$:

$$\vec{g_n}(t, t+l) = \begin{bmatrix} x_n(t+l) - x_n(t) \\ y_n(t+l) - y_n(t) \end{bmatrix}.$$

For each frame t, we compute three clusterings for the particles occurring in that frame based on the following features: (1) only the displacement vectors between adjacent frames ($\vec{g_n}(t, t+1)$); (2) only the displacement vectors with two-frames distances ($\vec{g_n}(t, t+2)$); and (3) only the three-frames distance displacement vectors ($\vec{g_n}(t, t+3)$).[1] These three clusterings are used in each frame to re-enforce the tendency of particles with similar displacement vectors $\vec{g_n}(t, t+1)$, $\vec{g_n}(t, t+2)$ and $\vec{g_n}(t, t+3)$ to group (i.e. to show similar motion patterns). This re-enforcement reduces the influence of outliers and clustering inconsistencies in the final particle clusters.

Considering that T is the number of frames, we will have $Q = 3T - 6$ clusterings of the set of particles.[2] Each clustering $\{\lambda^{(q)} | q \in \{1, ..., Q\}\}$ divides the set of particles into $k^{(q)}$ clusters $\{\chi_i^q | i \in \{1, ..., k^{(q)}\}\}$, where χ_i^q denotes the ith cluster of the qth clustering, and $k^{(q)}$ denotes the number of clusters in the qth clustering.

1 In each clustering, only the particle displacement vectors defined in the frame interval $[t, t+l]$, $l = \{1, 2, 3\}$, are considered.

2 Some displacement vectors $\vec{g_n}(t, t+l)$ can not be calculated: (a) $\vec{g_n}(t, t+1)$, $\vec{g_n}(t, t+2)$ and $\vec{g_n}(t, t+3)$ when $t = T$; (b) $\vec{g_n}(t, t+2)$ and $\vec{g_n}(t, t+3)$ when $t = T-1$; and (c) $\vec{g_n}(t, t+3)$ when $t = T-2$.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$
p_1	1	1	1
p_2	1	2	?
p_3	2	2	2
p_4	2	2	2
p_5	3	3	?
p_6	3	?	?

Table 1. Example of label vectors $\lambda^{(1,..,Q)}$, with $Q = 3$, $k^{(1)} = 3$, $k^{(2)} = 3$ and $k^{(3)} = 2$.

	$\mathbf{H}^{(1)}$			$\mathbf{H}^{(2)}$			$\mathbf{H}^{(3)}$	
	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
p_1	1	0	0	1	0	0	1	0
p_2	1	0	0	0	1	0	0	0
p_3	0	1	0	0	1	0	0	1
p_4	0	1	0	0	1	0	0	1
p_5	0	0	1	0	0	1	0	0
p_6	0	0	1	0	0	0	0	0

Table 2. Example of hypergraph with 8 hyperedges.

The mean-shift approach [2] was used to obtain the data clusterings $\lambda^{(q)}$ because of its ability to identify clusters with arbitrary shapes in the feature space. The bandwidth of the kernel density estimator of the mean-shift algorithm was set to $3 \cdot l$, for all frames t and $l = \{1, 2, 3\}$.

At this stage, we have three clusterings $\{\lambda^{(q)}\}$ per frame. Then, a unique partition is obtained for the full set of particles by merging the three clusterings per frame into a single clustering λ. To perform this task, we use the Meta-Clustering Algorithm (MCLA) [10].

First, the MCLA approach transforms the Q cluster label vectors $\lambda^{(q)}$ into a hypergraph. Recall that a hypergraph is constituted by nodes and hyperedges, and is a generalization of a graph, in the sense that a hyperedge can connect any set of nodes (while in a regular graph, an edge connects exactly two nodes). To each clustering $\lambda^{(q)} \in \mathbb{N}^n$ containing $k^{(q)}$ clusters, we assign a binary membership indicator matrix $\mathbf{H}^{(q)}$, with N rows (one for each particle) and $k^{(q)}$ columns (one for each cluster). In the hypergraph representation, nodes represent particles and the hyperedges represent clusters. Table 1 and 2 show for an illustrative hypergraph the label vectors of the clusterings $\lambda^{(q)}$ (see Table 1), and the corresponding hypergraph (see Table 2). In this example, the table rows represent the particles ($p_{\{1,...,6\}}$), the clusterings are represented by the label vectors $\lambda^{(1,\cdots,3)}$ (see Table 1), the respective binary membership indicator matrices are $\mathbf{H}^{(1,\cdots,3)}$ (see Table 2), and the individual clusters are represented by the hyperedges $h_{\{1,...,8\}}$. Since a particle can only belong to one cluster in a frame q, the row entries in the binary membership indicator matrix $\mathbf{H}^{(q)}$ add to '1' when the particle cluster is known (i.e. the particle is defined in the interval of frames $[t, t+l]$ used to produce clustering $\lambda^{(q)}$). Rows corresponding to particles not assigned to any cluster add to '0' (i.e., the corresponding particles do not exist in the interval of frames $[t, t + l]$ over which the clustering $\lambda^{(q)}$ was defined).

The concatenated block matrix $\mathbf{H} = \mathbf{H}^{(1,\cdots,Q)} = (\mathbf{H}^{(1)}...\mathbf{H}^{(Q)})$ defines the adjacency matrix of a hypergraph with N nodes (i.e. particles) and $\sum_{q=1}^{Q} k^{(q)}$ hyperedges (i.e. particle clusters). Each column vector h_a is a hyperedge of the hypergraph representation $\mathbf{H}$, and it represents one of the particle clusters of the set of clusters denoted by $\mathbf{H}^{(i)}$, $i = 1, ..., Q$. The MCLA method groups clusters (i.e. hyperedges) by similarity into meta-clusters. The final meta-clusters are obtained by assigning each particles inserted in the video to one meta-cluster.

The meta-clusters are obtained by hierarchically clustering hyperedges. To do that, we compute the symmetrical similarity matrix w, where $w(a, b) = w(b, a)$, as a similarity measure for any two hyperedges h_a and h_b of the hypergraph. The similarity between cluster pairs h_a and h_b is measured by the binary Jaccard measure:

$$w(a, b) = \frac{h'_a \cdot h_b}{||h_a||_2^2 + ||h_b||_2^2 - h'_a \cdot h_b} \tag{4}$$

Once the similarity matrix w has been calculated, the hyperedges are hierarchically clustered with the single link method [5]. The ideal number of meta-clusters K is chosen as the number of clusters that is the most *stable* in hierarchical clustering (i.e. dendrogram process). We observe how the number of clusters varies while the threshold values are increased when the dendrogram is built; K is the number of clusters detected during the longest range of consecutive increasing threshold values in the dendrogram for which the number of clusters does not change. This is illustrated in Figure 1, where the range Z in the dendrogram defines the ideal number of meta-clusters $K = 3$.

Identified the K-meta-clusters, all the hyperedges belonging to the same meta-cluster are merged into a single meta-hyperedge. This is done by averaging all hyperedges h_i belonging to the same meta-cluster. A particle can be associated to a different cluster in different frames, and the clusters to which a particle belongs can be associated to different meta-clusters (e.g. in some frames a particle may be in motion with respect to the background, but the same particle may be static in other frames). Averaging the hyperedges will result in meta-hyperedges that have an entry for each particle, describing the level of association of that particle with the corresponding meta-cluster. The stronger the

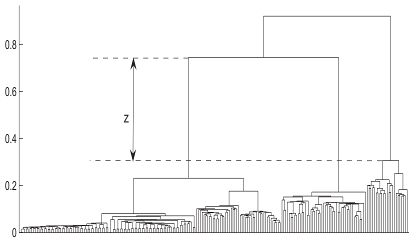


Figure 1. Dendrogram obtained through hierarchical decomposition using the single linkage method [5]. The longest range of thresholds values for which the number of clusters is constant (Z) determines the ideal number of clusters ($K = 3$).

association, the nearer to '1' the entry is; while the weaker the association, the nearer to '0' it gets.

The final step of the ensemble clustering process is the assignment of each particle to the meta-cluster with which it is associated most (i.e. has the highest entry in the association meta-hyperedge vector). At the end of this particle re-assignment by association with the K meta-clusters, a final set of meta-clusters is obtained $\{G_c | c \leq K\}$, and $c \leq K$ because some meta-clusters may have no particles associated with it. Thus, there will be at most K meta-cluster labels in the final particle clustering.

3.2. Particle Meta-Cluster Validation

As mentioned before, errors may occur in particle tracking, leading to particles wrongly clustered. To detect these inconsistent after the ensemble clustering step (see Section 3.1), a cluster validation step is performed by analyzing the trajectories in the neighborhood of the clustered particles.

Let $p_n(t) = (x_n(t), y_n(t))$ be the nth particle coordinates at the frame t. The context of this particle is defined by its neighborhood of size F, given by: $(x_n(t) + \delta_x, y_n(t) + \delta_y)$, where $-\frac{F-1}{2} \leq \delta_x, \delta_y \leq \frac{F-1}{2}$ and $\delta_x, \delta_y \in \mathbb{N}$. In all our experiments we used a neighborhood size $F = 5$.

The neighborhood of a particle in coherent motion should not change (since the whole ensemble is moving coherently). The differences occurring in the F-neighborhood of the particle n, between frame t and frame $t - L$, can be calculated based on the following matching errors:

$$\begin{aligned} &D^{[f]}(n, t, L) = \\ &\quad ||I(x_n(t) + \delta_x^{[f]}, y_n(t) + \delta_y^{[f]}, t) - \\ &\quad\; I(x_n(t - L) + \delta_x^{[f]}, y_n(t - L) + \delta_y^{[f]}, t - L)||_2, \end{aligned}$$

where $f = \{1, ..., F^2\}$ are indices to locations in the particle F-neighborhood. To estimate context changes, we compute the average of the k-best matches (i.e., the average of the k smallest $D^{[f]}(n, t, L)$ values in $f = \{1, ..., F^2\}$), and if this average value is large enough (i.e. larger than a threshold T_{best_k}, e.g. $T_{best_k} = 100$), we assume that the particle n changed its context at frame t, and now it can be marked as inconsistent. In our experiments, we found that $k = 15$ and $L = 3$ offers a good compromise between false positives and false negatives.

We use k-best neighborhood matches to reduce the influence of particles near to the boundaries of regions with coherent motion, since their context is affected by the other motions in adjacent regions. A small k value (in comparison with F^2) is chosen to allow slow transitions between adjacent regions (i.e. motion context changes), without identifying a change in context.

Let $\Omega_n = \{\omega_n^1, \omega_n^2, ..., \omega_n^{Z_n}\}$ be the sorted set of frame indices, indicating where possible context changes occurred for particle n. The lifetime of particle n is then sub-divided according to the intervals $\{[1, \omega_n^1), [\omega_n^1, \omega_n^2), ..., [\omega_n^{Z_n - 1}, \omega_n^{Z_n}), [\omega_n^{Z_n}, T]\}$. To determine if a context change actually occurred (and the particle trajectory estimate is incorrect), we keep track of motion and spatial location of the particle, comparing it with the cluster prototypes of these intervals. Then, the similarity between the motion pattern of particle n with respect to each cluster G_c is measured by:

$$S(n, c, t) = e^{-\left(\frac{d_m(n,c,t)}{2 \cdot \sigma_m^2} + \frac{d_s(n,c,t)}{2 \cdot \sigma_s^2}\right)} \tag{5}$$

where $d_m(n, c, t)$ and $d_s(n, c, t)$ denote the motion and space differences, respectively, between the particle n and the prototype of cluster G_c in frame t. The standard deviations σ_m and σ_s are set to '1'. The motion difference is defined as:

$$d_m(n, c, t) = \sqrt{(u_n(t) - u^{[c]}(t))^2 + (v_n(t) - v^{[c]}(t))^2}$$

where $u_n(t) = x_n(t) - x_n(t - 1)$ and $v_n(t) = y_n(t) - y_n(t - 1)$. $u^{[c]}(t)$ and $v^{[c]}(t)$ are respectively the horizontal and vertical components of the cluster G_c representative motion vector between the frames t and $t - 1$. This representative motion vector is calculated as follows:

1. Calculate the set of motion vectors $\{[u_n(t), v_n(t)] | p_n \subset G_c\}$;
2. Compute the reduced ordering of the respective motion vectors, in relation to the base vector: $[\texttt{min}(u_n(t)), \texttt{min}(v_n(t))]$;
3. The vector corresponding to the median value in the reduced ordering is assigned to $[u^{[c]}(t), v^{[c]}(t)]$.

The space difference measure is defined by:

$$d_s(n,c,t) = \frac{1}{k} \cdot \sum_{j=1}^{k} \sqrt{(x_n(t) - x_{n,j}^{[c]}(t))^2 + (y_n(t) - y_{n,j}^{[c]}(t))^2}$$

where $\{(x_{n,j}^{[c]}(t), y_{n,j}^{[c]}(t)) | j = 1, ..., k\}$ are the spatial coordinates of the k-nearest particles belonging to cluster G_c in relation to particle n in frame t.

To determine to which cluster a particle n is most likely to belong within an interval $[\omega_n^1, \omega_n^2)$, we assign the particle to the cluster that maximizes the similarity measure below (see Eq. 5) in the interval:

$$c_{max} = \max_c \sum_{t=\omega_n^1}^{\omega_n^2 - 1} S(n,c,t) \qquad (6)$$

In order to have a consistent representation for particles and particle clustering, we define that a particle can belong only to one cluster in its entire lifetime. Thus, when a particle is assigned to more than one cluster by the labeling criterion in Eq. 6, this particle is split into as many particles as are the cluster labels it was assigned to.

Note that a possible context change detection not necessarily implies a cluster change after the cluster re-assignment. A particle re-classification only occurs when the context modification is associated with spatial/motion incoherence.

3.3. Spatial Filtering

The last stage of our particle classification process is the spatial particle filtering. The goal of spatial filtering is to eliminate outliers and groups of adjacent particles that are not significant (artifacts).

To represent spatial adjacency of particles, we compute the Delaunay triangulation $DT(t)$ for all particles locations $(x_n(t), y_n(t))$ on frame t. Two particles are considered adjacent if they share an edge of $DT(t)$. The membership of adjacent particles are represented by assigning binary weights to the edges of $DT(t)$, i.e. an edge receives '1' if it connects two particles belonging to the same cluster; or it receives '0' if the particles belong to distinct clusters. We examine all the connected components of $DT(t)$ [3] and re-assign to other clusters all the particles belonging to clusters that contain less than 20 nodes. These particles are assigned to the cluster that share more '0' weight edges with the cluster that had its particles re-assigned.

3 Two particles are in the same connected component if and only if there is some path between them composed only by edges of weight '1'.

4. Experiments and Results

This section presents some experimental results obtained with the proposed method. The algorithm described in this paper was applied to a sequence of 50 frames of the *coastguard* sequence, and Figure 2 illustrates the obtained results for three frames of the sequence. The particle sampling and tracking process resulted in 30612 particles over the 50 frame sequence, with an average particle lifetime of approximately 24 frames. The first column in Figure 2 shows the frames 5, 30 and 45 of the sequence. The second column shows the three initial particle clusterings (as obtained by mean-shift), where the colors of particles indicate the cluster labels for that frame. The third column depicts the results obtained after meta-clustering, with $K = 3$ meta-clusters (the correspondent meta-clustering dendrogram appears in Figure 1). The fourth column shows the final results obtained for the frames 5, 30 and 45 of the sequence (after cluster validation and spatial filtering).

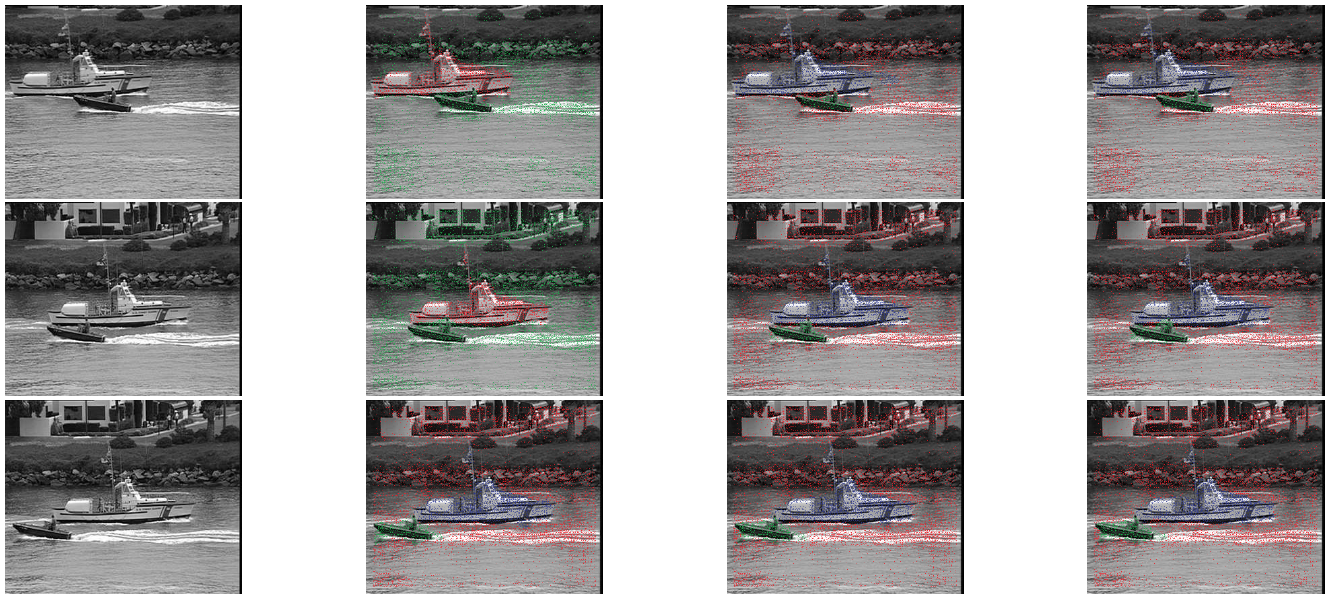
5. Conclusions

A method for unsupervised identification of coherent motion in video sequences has been proposed in this paper. This technique provides a new way of linking raw video data to high-level (abstract) concepts, as required in many image processing and computer vision tasks. Some of its advantages and limitations are discussed next.

Since the first clustering stage groups particles with similar displacement vectors, it constrains the coherent motion identification to rigid-body translational motion only. However, this is a soft constraint, since the second clustering stage (ensemble clustering) identifies the tendency along the entire video and re-groups the particles. Thus, the proposed method can identify any type of coherent motion, as long as the motion is approximately rigid-body translational between adjacent video frames.

Although in the present work we used a hard clustering method to obtain the individual particle clusterings (i.e. mean-shift), it is also possible to use soft clustering in the same framework. This can be particularly useful in applications where models of object motions are available, and probabilistic clustering can be used. In this case, the Jaccard measure (see Equation 4) could be used with non-binary vectors, with each element representing the probability that the respective particle belongs to a cluster, or could be replaced with some other vector similarity measure.

The clustering approach used in this work has a significant disadvantage: it requires the definition of a fixed bandwidth for mean-shift clustering (see Section 3.1). The ideal bandwidth value depends on the type of motion to be detected in the video sequence, and different motion types may occur in the same video sequence.



(a) (b) (c) (d)

Figure 2. Coastguard sequence: (a) original frames, (b) mean-shift clustering, (c) meta-clusters, (d) final results. First row: frame 5; Second row: frame 30; Third row: frame 45.

As future work, we plan to further develop our approach by investigating how to model moving structures, and how to represent object occlusions in video scenes.

Acknowledgments

The authors would like to thank Dr. Peter Sand for providing the optical flow code. This research was partially supported by CNPq - Brazilian National Research Council.

References

[1] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision (ECCV)*, volume 3024 of *LNCS*, pages 25–36, Prague, Czech Republic, May 2004. Springer.

[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.

[3] C. E. Erdem, B. Sankur, and A. M. Tekalp. Video object tracking with feedback of performance measures. *IEEE Trans. Circuits Syst. Video Techn.*, 13(4):310–324, 2003.

[4] A. L. N. Fred and A. K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(6):835–850, 2005.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[6] M. Ristivojevic and J. Konrad. Space-time image sequence analysis: object tunnels and occlusion volumes. *IEEE Transactions on Image Processing*, 15(2):364–376, 2006.

[7] P. Sand. *Long-Range Video Motion Estimation using Point Trajectories*. PhD thesis, Massachusetts Institute of Technology, July 2006.

[8] P. Sand and S. J. Teller. Particle video: Long-range motion estimation using point trajectories. In *CVPR (2)*, pages 2195–2202, 2006.

[9] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[10] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, 2003.

[11] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Comput. Vision*, 9(2):137–154, 1992.

[12] P. Viswanath and K. Jayasurya. A fast and efficient ensemble clustering method. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 720–723, Washington, DC, USA, 2006. IEEE Computer Society.

[13] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *ECCV (4)*, pages 94–106, 2006.

[14] J. Zhang, J. Gao, and W. Liu. Image sequence segmentation using 3-d structure tensor and curve evolution. *IEEE Trans. Circuits Syst. Video Techn.*, 11(5):629–641, 2001.