Camera Motion Estimation Using a Novel Online Vector Field Model in Particle Filters

Symeon Nikitidis†, Stefanos Zafeiriou† and Ioannis Pitas†, Fellow Member, IEEE

***Abstract*— In this paper, a novel algorithm for parametric camera motion estimation is introduced. More particularly, a novel stochastic vector field model is proposed, which can handle smooth motion patterns derived from long periods of stable camera motion and can also cope with rapid camera motion changes and periods when the camera remains still. The stochastic vector field model is established from a set of noisy measurements, such as motion vectors derived e.g. from block matching techniques, in order to provide an estimation of the subsequent camera motion in the form of a motion vector field. A set of rules for a robust and online update of the camera motion model parameters is also proposed, based on the Expectation Maximization algorithm. The proposed model is embedded in a particle filters framework, in order to predict the future camera motion based on current and prior observations. We estimate the subsequent camera motion by finding the optimum affine transform parameters so that, when applied to the current video frame, the resulting motion vector field to approximate the one estimated by the stochastic model. Extensive experimental results verify the usefulness of the proposed scheme in camera motion pattern classification and in the accurate estimation of the 2D affine camera transform motion parameters. Moreover, the camera motion estimation has been incorporated into an object tracker in order to investigate if the new schema improves its tracking efficiency, when camera motion and tracked object motion are combined.**

***Index Terms*— Camera Motion Estimation, Expectation Maximization Algorithm, Particle Filtering, Vector Field Model**

I. Introduction

Motion estimation and motion pattern classification produce valuable information for video processing, analysis, indexing and retrieval.It has been extensively investigated by the scientific community for semantic characterization and discrimination of video streams. Moving object trajectories have been used for video retrieval [1]-[3]. Camera motion pattern characterization has been efficiently applied to video indexing and retrieval [4]-[7]. However, the main limitation of the latter methods, is that they deal only with the characterization of the detected camera motion patterns, without explicit measurement of the camera motion parameters. As a result, the acquired information is of limited interest, since it can be used primarily for video indexing and retrieval.

The estimation of a parametric form describing the displacement of the video frame content in two subsequent video frames due to camera motion is of broader interest and has been proved beneficial in various applications. For instance, camera motion parameter estimation can assist in detecting and robustly tracking moving objects [8], in motion-based video deblurring [9], in video shots boundaries detection [10], as well as, in video abstraction [11],[12]. Apart from the general 2-D affine transformation model (that we have also adopted in this paper) various parametric camera motion representation methods have been proposed in the literature n [4], [13], [14], [15], [16], [17], [18], [19].

In this paper, we focus on the 2D camera motion characterization and the estimation of the relevant affine parametric model. Various methods have been proposed to this end, by exploiting estimated motion vector fields. In [5], the motion vectors field is used as a camera motion representation and the detected motion pattern is classified using Support Vector Machines (SVMs) in one of the following classes: zoom, pan, tilt and rotation. In [4], [6] and [18] camera motion estimation within video shots is performed in the compressed MPEG video streams, without full frame decompression, using the motion vector fields acquired from the P- and B- video frames. These methods rely on the exploitation of motion vectors distribution or on a few representative global motion parameters. The detected camera motion is then expressed in a parametric form and is applied for video frame annotation and retrieval. One of the main shortcomings of these approaches is that, generally, they are neither resilient to the presence of moving objects of significant size nor to video luminance outliers.

In this work, we focus on accurate camera motion parameter estimation using already estimated motion vectors fields. In this approach, we assume the camera motion as a dynamic system, whose state $\boldsymbol{\theta}_t$ changes in discrete time intervals and is described at time t by the state vector:

$$\boldsymbol{\theta}_t = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 & m_5 & m_6 \end{bmatrix}^T, \qquad (1)$$

where parameters $\{m_1, m_2, m_3, m_4, m_5, m_6\}$ correspond to the affine transform coefficients, containing all the relevant information required to describe the camera motion between video frames. A novel stochastic vector field model is established from a set of noisy measurements $\mathbf{Y}_t$, such as the estimated motion vectors, in order to provide an estimation of the subsequent camera motion. Our goal is to recursively estimate the optimal affine transform parameters, by estimating the system state vector $\boldsymbol{\theta}_t$, so that, when applied to the current video frame, the resulting transformed image provided by a motion compensation algorithm accurately recreates the already estimated motion vectors field.

To tackle this problem, we have applied the proposed stochastic vector field model in a particle filters framework. Particle filters are a state-of-the-art method for the stochastic prediction of dynamic system state. Stochastic approaches used for the prediction of the future state of a dynamic system have attracted considerable interest against their deterministic

counterparts. Their ability to escape from local minima due to the fact that the search operation is randomly driven, is a significant advantage. However, the computational load is generally more intense compared with that of a deterministic algorithm. In summary, the novel contributions of this paper are the following:

- The presentation of a system for 2D camera motion estimation from a video sequence, that is able to perform in real time.
- A novel stochastic vector field model. The proposed model can handle smooth camera motion patterns derived from long periods of stable camera motion and can also cope with rapid motion changes (i.e., motion changes from hand handled cameras) and periods when the camera remains still.
- An online Expectation Maximization (EM) algorithm for updating the model parameters.

The rest of the paper is organized as follows. Section II we provide an estimation of the affine parametric model based on the straightforward minimization of the Least Squares Error between the motion fields. The proposed Online Vector Field Model and the applied particle filters framework, are presented in Section III. In the same Section, considerations on enhancing its algorithmic performance and achieving computational efficiency are analyzed. Section IV describes the conducted experiments and summarizes the performance evaluation results. Concluding remarks are drawn in Section V.

II. Problem Formulation

Initially, we present a camera motion estimation model that translates the motion vector field derived from two consecutive video frames into a parametric 2D affine transform. The 2D affine transformation of an image point displaced from position (x, y) to (x', y') between two consecutive video frames is given by:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

where the parameters $\{m_1, m_2, m_4, m_5\}$ control rotation and scaling, while parameters $\{m_3, m_6\}$ correspond to translation along x and y axes, respectively.

We address the camera motion detection and estimation problem by employing low level information such as motion vectors. We detect the motion vectors between two successive video frames by applying a motion estimation algorithm, such as block matching and represent the detected displacements using motion vectors. A motion vector $\mathbf{v}^i = [v_x^i \quad v_y^i]^T$ represents the displacement of the i-th block in relative coordinates, with respect to its initial position, between two consecutive video frames f_{t-1} and f_t as: $x_i' = x_i + v_x^i$, $y_i' = y_i + v_y^i$ where (x_i, y_i) and (x_i', y_i') are the coordinates of i-th block center at frame f_{t-1} and f_t respectively.

Similarly, with the image point displacement described by (2), we can represent the displacement of the i-th block in relative coordinates by a 2D affine transform as:

$$\begin{bmatrix} v_x^i \\ v_y^i \\ 0 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \tag{3}$$

where the affine coefficients are related as: $[c_1\ c_2\ c_3\ c_4\ c_5\ c_6]^T = [m_1 - 1\ m_2\ m_3\ m_4\ m_5 - 1\ m_6]^T$.

Since we estimate the camera affine transformation by utilizing the motion vector field, a model to compute the affine transform coefficients directly from the motion vectors, is required. To rephrase the problem, we seek an affine transformation matrix $\mathbf{M}$ to perform the approximation $\mathbf{BM} \approx \mathbf{B}+\mathbf{V}$, where $\mathbf{B}$ is a $n \times 3$ matrix (n is the number of blocks that each frame has been divided to) containing the center coordinates of each block in the video frame, i.e. $\mathbf{B} = [\mathbf{b}_x\ \mathbf{b}_y\ \mathbf{1}]$, where $\mathbf{b}_x = [x_1\ x_2 \dots x_n]^T$ and $\mathbf{b}_y = [y_1\ y_2 \dots y_n]^T$. The matrix $\mathbf{V} = [\mathbf{v}_x\ \mathbf{v}_y\ \mathbf{0}]$ contains the motion vectors, where $\mathbf{v}_x = [v_x^1 \quad v_x^2 \dots v_x^n]^T$ and $\mathbf{v}_y = [v_y^1 \quad v_y^2 \dots v_y^n]^T$ are $n \times 1$ vectors containing each block's displacement in relative coordinates along to the x and y axes, respectively. $\mathbf{M} = [\mathbf{M}_x\ \mathbf{M}_y\ \mathbf{e}]$ is the 3×3 affine transformation matrix, where $\mathbf{M}_x = [m_1 \quad m_2 \quad m_3]^T$, $\mathbf{M}_y = [m_4 \quad m_5 \quad m_6]^T$ and $\mathbf{e} = [0 \quad 0 \quad 1]^T$.

We have experimentally verified that it is preferable to seek independently the vectors $\mathbf{M}_x$ and $\mathbf{M}_y$, rather than to search directly for the matrix $\mathbf{M} = [\mathbf{M}_x \quad \mathbf{M}_y \quad \mathbf{e}]$. The LS formulation for the optimal $\mathbf{M}_{x,o}$ takes the form:

$$\mathbf{M}_{x,o} = \min_{\mathbf{M}_x} G(\mathbf{b}_x + \mathbf{v}_x, \mathbf{BM}_x) \tag{4}$$

where $G(\mathbf{A}, \mathbf{B}) = ||\mathbf{A}-\mathbf{B}||_F^2$ and $||.||_F$ is the Frobenius norm. The optimal $\mathbf{M}_{x,o}$ is given by:

$$\mathbf{M}_{x,o} = \mathbf{B}^+(\mathbf{b}_x + \mathbf{v}_x) \tag{5}$$

and similarly for $\mathbf{M}_{y,o}$.

According to (5), we can compute the affine transform coefficients describing the camera motion directly from the motion vector field, since the pseudo-inverse matrix $\mathbf{B}^+$ remains constant. This technique, whilst being optimal for data contaminated by Gaussian noise, is extremely inaccurate in the presence of motion vector outliers.

III. Online Vector Field Model

A. Basic Nomenclature for the Proposed Framework

$\boldsymbol{\theta}_t$	The unknown state of the dynamical system.
$\mathbf{v}_t^j$	The motion vector $\mathbf{v}_t^j$ at the t-th time (frame) of the j-th block.
$\mathbf{Y}_t$, $\hat{\mathbf{Y}}_t^i$	The observation (in our case a matrix of all the motion vectors $\mathbf{Y}_t = [\mathbf{v}_t^1 \dots \mathbf{v}_t^n]$) and the estimate produced by the i-th particle.
$\mathbf{Y}_{1:t}$	The set of observations in a time window between 1 and t.
$E_{t-1}(.,.)$	The system function that calculates the unknown state of the dynamical system.
$O_t(.,.)$	The system function that calculates the observations at time t.
P_t	A particle $P_t = \{\hat{\boldsymbol{\theta}}_t, w_t\}$.

$P(A|B)$ The conditional probability of the event A given the event B.

U_{t-1} The system noise at $t-1$ (it can be a vector or a matrix).

N_t The observation noise at t.

w_t^i The weight of i-th particle filter at time t.

$\mathbb{S}_t$ The stable component $\mathbb{S}_t = \{\mathbf{S}_{t,x}, \mathbf{S}_{t,y}\}$.

$\mathbf{S}_{t,x}$ $\mathbf{S}_{t,x} = [s_{t,x}^1 \quad s_{t,x}^2 \dots s_{t,x}^n]^T$ for the x-coordinate, at the t-th frame and for the n-blocks.

$\mathbf{S}_{t,y}$ $\mathbf{S}_{t,y} = [s_{t,y}^1 \quad s_{t,y}^2 \dots s_{t,y}^n]^T$ for the y-coordinate, at the t-th frame and for the n-blocks.

$\mathbb{W}_t$ The wander component $\mathbb{W}_t = \{\mathbf{W}_{t,x}, \mathbf{W}_{t,y}\}$.

$\mathbf{W}_{t,x}$ $\mathbf{W}_{t,x} = [w_{t,x}^1 \quad w_{t,x}^2 \dots w_{t,x}^n]^T$ for the x-coordinate, at the t-th frame and for the n-blocks.

$\mathbf{W}_{t,y}$ $\mathbf{W}_{t,y} = [w_{t,y}^1 \quad w_{t,y}^2 \dots w_{t,y}^n]^T$ for the y-coordinate, at the t-th frame and for the n-blocks.

$\mathbb{L}_t$ The lost component $\mathbb{L}_t = \{\mathbf{L}_{t,x}, \mathbf{L}_{t,y}\}$

$\mathbf{L}_{t,x}$ $\mathbf{L}_{t,x} = [l_{t,x}^1 \quad l_{t,x}^2 \dots l_{t,x}^n]^T$ for the x-coordinate, at the t-th frame and for the n-blocks.

$\mathbf{L}_{t,y}$ $\mathbf{L}_{t,y} = [l_{t,y}^1 \quad l_{t,y}^2 \dots l_{t,y}^n]^T$ for the y-coordinate, at the t-th frame and for the n-blocks.

$\boldsymbol{\mu}_{c,t}^j$ The mean motion vector for $c \in \{\mathbb{S}_t, \mathbb{W}_t, \mathbb{L}_t\}$ for the stable, the wander and the lost component for the j-th block at the t-th frame.

$\boldsymbol{\Sigma}_{c,t}^j$ The covariance matrix for motion vectors of the j-th block for one of the components $c \in \{\mathbb{S}_t, \mathbb{W}_t, \mathbb{L}_t\}$ at the t-th frame.

$P_u(.)$ The probability density function of Gaussian noise U_{t-1}.

$F_t(k)$ The exponential envelope.

$O_{c,t,xy}^j, O_{c,t,x}^j, O_{c,t,y}^j$ Ownerships for one of the components $c \in \{\mathbb{S}_t, \mathbb{W}_t, \mathbb{L}_t\}$ for the j-th block at instance t and for the xy, x and y, respectively.

$M_{1,t,x}^j, M_{1,t,y}^j, M_{2,t,x}^j, M_{2,t,y}^j$ First and second order moments for the j-th block at x and y coordinates, respectively for the t-th frame (only for stable component).

$m_{c,t,xy}^j, m_{c,t,x}^j, m_{c,t,y}^j$ Mixture weights for one of the components $c \in \{\mathbb{S}_t, \mathbb{W}_t, \mathbb{L}_t\}$ for the j-th block at instance t and for the xy, x and y, respectively.

$\hat{A}$ Denotes the estimate of A (e.g., $\hat{\boldsymbol{\theta}}_t$ denotes the estimate of $\boldsymbol{\theta}_t$).

$\hat{\boldsymbol{\mu}}_t$ Robust estimate for mean motion vector at the t instance.

$\hat{\boldsymbol{\Sigma}}_t$ Robust estimate for the covariance matrix for the motion vectors at t-th instance.

B. Particle Filters

Considering camera motion as a dynamically varying system, we formulate the problem as to predict the unknown state $\boldsymbol{\theta}_t$ based on a series of usually noisy motion observations (already estimated motion vectors) $\mathbf{Y}_{1:t} = \{\mathbf{Y}_1, \dots, \mathbf{Y}_t\}$, arriving sequentially. Moreover, we assume that the state evolution and observation models are described by the functions $E_{t-1}(.,.)$ and $O_t(.,.)$ respectively:

$$\boldsymbol{\theta}_t = E_{t-1}(\boldsymbol{\theta}_{t-1}, U_{t-1}) \tag{6}$$

$$\mathbf{Y}_t = O_t(\boldsymbol{\theta}_t, N_t), \tag{7}$$

where U_{t-1} is the system noise and N_t is the observation noise.

A particle is a weighted sample that estimates a required posterior density function [20],[21]. In the state evolution problem summarized in (6), a particle $P_t = \{\hat{\boldsymbol{\theta}}_t, w_t\}$ describes at time t the posterior distribution $P(\boldsymbol{\theta}_t|\mathbf{Y}_{1:t})$, where the weight w_t is normalized and is proportional to the posterior probability $P(\mathbf{Y}_t|\hat{\boldsymbol{\theta}}_t)$. To initialize a particle filters framework, we first draw K samples $\{\boldsymbol{\theta}_{t-1}^i\}_{i=1}^K$ from $P(\boldsymbol{\theta}_{t-1}|\mathbf{Y}_{1:t-1})$ and additionally, K samples $\{U_{t-1}^i\}_{i=1}^K$ from U_{t-1}. By applying state evolution as formed in (6), we obtain K estimates $\{\hat{\boldsymbol{\theta}}_t^i\}_{i=1}^K$ of state $\boldsymbol{\theta}_t$, which are being fed along with K noise samples $\{N_t^i\}_{i=1}^K$, drawn from N_t, to (7). Finally, K observation estimates $\{\hat{\mathbf{Y}}_t^i\}_{i=1}^K$ for state $\boldsymbol{\theta}_t$ are obtained. Each particle's weight is being evaluated with respect to the state and observation estimations, according to the formula:

$$w_t^i \propto \frac{P(\mathbf{Y}_t^i|\boldsymbol{\theta}_t)P(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})}{g(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{Y}_{1:t}),} \tag{8}$$

where $g(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{Y}_{1:t})$ is a proposal distribution. In order the weights to sum up to one, they are normalized as:

$$w_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i} \tag{9}$$

Here we model each posterior distribution $P(\hat{\mathbf{Y}}_t^i|\hat{\boldsymbol{\theta}}_t^i)$ using a mixture of bivariate Gaussian density functions, while we assume that noise U_{t-1} and N_t are also Gaussian.

C. Probabilistic Mixture Model

The proposed Online Vector Field Model ($OVFM_t$) is a modified version of the Online Appearance Model (OAM) by Jepson *et al.* presented in [22]. OAM is a three parts mixture model containing the following components: *the stable component* designed such as to identify slowly varying robust appearance properties of the tracked object, *the wandering component* that models the rapid variations of the object appearance and the *lost component* designed to handle data outliers that burst during occlusion. The first two components have been designed to follow the Gaussian distribution, while the data modelled by the lost component are assumed to be uniformly distributed.

We modified this model so as to facilitate camera motion estimation. The proposed $OVFM_t$ is a Gaussian mixture model extending the notion of stable or rapidly changing image structures of OAM to the description of the motion vector field. Thus, we can identify not only reliable motion structures but also rapid motion changes, as well. Moreover, we have modified the lost component, so as to represent the ideal stationary scene in order to adjust the model to have a prior preference in generating stationary camera motion estimations in the presence of data outliers, as for instance, due to motion vectors generated by moving objects. Additionally,

we handle motion outliers using robust statistics. Finally, we combined the $OVFM_t$ in a particle filter framework to estimate the camera motion parameters, while in [22] the EM algorithm is used in order to perform object tracking.

The previously presented fundamental types of 2D camera motion (in Section II) could be combined or appear in cascade in a video sequence. However, there are temporal camera motion characteristics that could be exploited in a camera motion estimation model. For example, in a typical video sequence, we expect long periods of smooth camera motion towards a specific motion direction which are also followed by extended camera immobility periods. These two motion patterns are usually interrupted by brief time intervals of rapid camera motion that could be of arbitrary type. For the first two motion patterns, a model that identifies slowly varying (or absence of) motion observations over a long period of time is more appropriate for their description. In the latter case, when the camera is rapidly and arbitrary moving, a flexible model based on two video frame variations can better approximate the rapid changes.

In the presented $OVFM_t$, we use the motion vector field derived either by applying of block matching algorithm or directly from compressed MPEG video streams. The model is time-varying and comprises of three different components $OVFM_t = \{\mathbb{S}_t, \mathbb{W}_t, \mathbb{L}_t\}$, which are combined in a probabilistic mixture model applied in a particle filters framework, in order to estimate the camera motion.

- The camera motion stable component $\mathbb{S}_t = \{\mathbf{S}_{t,x}, \mathbf{S}_{t,y}\}$ learns a smooth motion pattern that describes the camera motion obtained from a relatively long period of the video sequence. The component $\mathbb{S}_t$ comprises of the vectors $\mathbf{S}_{t,x} = [s^1_{t,x} \quad s^2_{t,x} \dots s^n_{t,x}]^T$ and $\mathbf{S}_{t,y} = [s^1_{t,y} \quad s^2_{t,y} \dots s^n_{t,y}]^T$, where values $s^j_{t,x}$ and $s^j_{t,y}$ contain the block j spatial displacement of time t smoothed over a predefined time window along the x and y axes, respectively: $s^j_{t,x} = \lambda v^j_{t,x} + (1-\lambda) s^j_{t-1,x}$ $s^j_{t,y} = \lambda v^j_{t,y} + (1-\lambda) s^j_{t-1,y}$ where the smoothing factor λ is proportional to the temporal window size (measured in video frames), $v^j_{t,x}$ and $v^j_{t,y}$ are respectively the x and y motion vector components referred to the j-th block.
- Since the component $\mathbb{S}_t$ requires a long sequence of observations in order to construct a smoothed camera motion vector field, we cannot have a good approximation when severe camera motion changes occur. In order to address this problem, we introduce the camera motion wander component $\mathbb{W}_t = \{\mathbf{W}_{t,x}, \mathbf{W}_{t,y}\}$, which identifies sudden motion changes, and adapts to a short time motion field observation sequence, as a two frame motion change model. Vectors $\mathbf{W}_{t,x} = [w^1_{t,x} \quad w^2_{t,x} \dots w^n_{t,x}]^T$ and $\mathbf{W}_{t,y} = [w^1_{t,y} \quad w^2_{t,y} \dots w^n_{t,y}]^T$ contain each block displacement between two consecutive frames, in relative coordinates, along the x and y axes, respectively.
- Finally, the lost component $\mathbb{L}_t = \{\mathbf{L}_{t,x}, \mathbf{L}_{t,y}\}$ is fixed and represents the ideal stationary video scene when all the motion vectors are equal to zero. This is the state that is expected to be observed more often. Moreover, it is used for the initialization of a new camera motion estimation process and also enables the model to have a prior preference in generating stationary camera motion estimations when sparse non zero motion vectors are observed, as for instance, due to objects motion.

We model the probability density function for the $\mathbb{S}_t$, $\mathbb{W}_t$ and $\mathbb{L}_t$ components with the bivariate Gaussian distribution $N(\mathbf{v}^j; \boldsymbol{\mu}^j_{c,t}, \boldsymbol{\Sigma}^j_{c,t})$ $c \in \{\mathbb{S}_t, \mathbb{W}_t, \mathbb{L}_t\}$, where $\boldsymbol{\mu}^j_{c,t}$ denotes the mean value of the j-th motion vector and $\boldsymbol{\Sigma}^j_{c,t}$ is a 2×2 covariance matrix referred to c-th component j-th motion vector $\mathbf{v}^j = [v^j_x \quad v^j_y]^T$, as it varies during the video sequence. We consider the general case that correlation exists between the two random variables v^j_x, v^j_y of the same motion vector $\mathbf{v}^j$, as it evolves over time. It should be noted that the stable component covariance matrices $\boldsymbol{\Sigma}^j_{\mathbb{S},t}$ and mean values $\boldsymbol{\mu}^j_{\mathbb{S},t}$ are functions of time computed for each motion vector. The wander component, the mean values are the observations of the previous frame and for $\mathbb{L}_t$ the mean values are set to zero. Moreover, in order to avoid some prior preference in either component, the covariance matrices are initially set as: $\hat{\boldsymbol{\Sigma}}^j_{\mathbb{S},t} = \hat{\boldsymbol{\Sigma}}^j_{\mathbb{W},t} = \hat{\boldsymbol{\Sigma}}^j_{\mathbb{L},t}$ (more details are given in Model Initialization subsection).

$OVFM_t$ combines probabilistically the components $\mathbb{S}_t$, $\mathbb{W}_t$ and $\mathbb{L}_t$ in a mixture model according to the formula:

$$\begin{aligned} P \quad & (\mathbf{Y}_t|\boldsymbol{\theta}_t) \\ & = \prod_{j=1}^n \Big\{ P(\mathbf{v}^j_t|\mathbb{S}^j_t) + P(\mathbf{v}^j_t|\mathbb{W}^j_t) + P(\mathbf{v}^j_t|\mathbb{L}^j_t) \Big\} \\ & = \prod_{j=1}^n \Big\{ \textstyle\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} m^j_{c,t,xy} N\Big(\mathbf{v}^j_t; \boldsymbol{\mu}^j_{c,t}, \boldsymbol{\Sigma}^j_{c,t}\Big) \Big\}, \end{aligned} \tag{10}$$

where $\mathbf{Y}_t = [\mathbf{v}^1_t \dots \mathbf{v}^n_t]^T$ is the observation data derived for state $\boldsymbol{\theta}_t$ and $N(\mathbf{v}^j_t; \boldsymbol{\mu}^j_{c,t}, \boldsymbol{\Sigma}^j_{c,t})$ is the bivariate Gaussian density function:

$$N(\mathbf{v}^j_t; \boldsymbol{\mu}^j_{c,t}, \boldsymbol{\Sigma}^j_{c,t}) = \frac{1}{2\pi\sqrt{|\boldsymbol{\Sigma}^j_{c,t}|}} e^{-\frac{1}{2}(\mathbf{v}^j_t - \boldsymbol{\mu}^j_{c,t})^T (\boldsymbol{\Sigma}^j_{c,t})^{-1} (\mathbf{v}^j_t - \boldsymbol{\mu}^j_{c,t})} \tag{11}$$

where $\mathbf{v}^j_t = [v^j_{t,x} \; v^j_{t,y}]^T$, $\boldsymbol{\mu}^j_{c,t} = [\mu^j_{c,t,x} \; \mu^j_{c,t,y}]^T$,$\mathbf{m}^j_{c,t}[m^j_{c,t,x} \; m^j_{c,t,y} \; m^j_{c,t,xy}]^T$,
$\boldsymbol{\Sigma}^j_{c,t} = \begin{bmatrix} (\sigma^j_{c,t,x})^2 & (\sigma^j_{c,t,xy})^2 \\ (\sigma^j_{c,t,xy})^2 & (\sigma^j_{c,t,y})^2 \end{bmatrix}$, $c \in \{\mathbb{S}, \mathbb{W}, \mathbb{L}\}$.

$\{m^j_{\mathbb{S},t,xy}, \; m^j_{\mathbb{W},t,xy}, \; m^j_{\mathbb{L},t,xy}\}$ are the mixing probabilities that regulate the contribution each component j-th motion vector makes to the complete observation likelihood at time t, n is the number of motion vectors, $\boldsymbol{\Sigma}^j_{c,t}$ and $\boldsymbol{\mu}^j_{c,t}$ are the covariance matrix and mean value, respectively, referred to the j-th motion vector of the c-th component.

$OVFM_t$ is embedded in the particle filter framework evaluating each potential future state of the system. A state estimate $\hat{\boldsymbol{\theta}}^i_t$ is generated by first drawing a noise sample $U^i_{t-1} \sim P_u(U_{t-1})$ and applying the state transition function $\hat{\boldsymbol{\theta}}^i_t = E_{t-1}(\boldsymbol{\theta}^i_{t-1}, U^i_{t-1})$ where $P_u(.)$ is the probability density function of Gaussian noise U_{t-1}. Each state estimate $\hat{\boldsymbol{\theta}}^i_t$ determined by particle i is being evaluated with respect to the available motion representation in $OVFM_t$, by computing the observation likelihood according to (10). Although the conventional particle filters configuration determines the particle weight using (8), we instead update the weights by

applying the an approach similar to the Sequential Importance Re-sampling filter (SIR) [23], since the following assumptions hold:

- The state evolution E_{t-1} and observation O_t functions are known.
- The observation likelihood function $P(\mathbf{Y}_t|\boldsymbol{\theta}_t)$ could be applied for pointwise evaluation.

As a result, we assign weights to particles as:

$$\begin{aligned} w_t^i &\propto P(\mathbf{Y}_t^i|\boldsymbol{\theta}_t^i), \\ w_t^i &= \tfrac{w_t^i}{\sum_{i=1}^N w_t^i}, \end{aligned} \tag{12}$$

this basically dropping the factors $p(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ and $g(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{Y}_{1:t})$. The applied particle filters framework is similar to the one used in [24]. The particle filters framework generates a set of possible future states of the camera motion, expressed in the form of affine transform matrices. Each affine transform matrix corresponds to a motion vector field which can be computed using equation (5). Consequently, each state estimate is evaluated with respect to the available motion representation in $OVFM_t$ via equation (10) and is assigned a weight according to (12). The particle that is assigned the highest weight or differently the prediction that achieves the highest probability value is selected as the system's future state.

D. Online Model Update

In order to update the camera motion mixture model $OVFM_t$ to $OVFM_{t+1}$, describing the camera motion mixture model in the next video frame the new mean values, covariance matrices and mixture probabilities for each motion vector contained in each component at time $t+1$ should be estimated. We assume that $OVFM_t$ has limited memory over the past motion vector field observations, extended during a defined time window, which is exponentially forgotten. When newer information is available, previous knowledge is forgotten and is combined with newer observations. The exponential envelop $F_t(k) = \alpha e^{(-(t-k)/\tau)}$ for $k \leqslant t$ is being used where $\tau = n_s/\log 2$ and n_s is the envelope half life time, measured in video frames that the current information is preserved in the system's memory. This information exponentially weakens during time and completely vanishes after a predefined time window. Thus, parameter $F_t(k)$ is used in order to regulate the influence of prior knowledge. Parameter α is defined as $\alpha = 1 - e^{-1/\tau}$, so that the envelop weights $F_t(k)$ sum to 1. The new mixing and ownership posterior probabilities, the mean values and covariance matrices for each motion vector of the $\mathbb{S}_t$ and $\mathbb{W}_t$ components are being updated with respect to the envelop weights $F_t(k)$.

The posterior ownership probabilities $O_{c,t}^j$ denote the contribution of each motion vector to the complete observation probability likelihood function. We favor these motion vectors that continuously produce higher probability values by increasing their ownership probability. On the other hand, motion vectors that tend to produce lower probability values are penalized and their contribution to the complete observation likelihood is gradually reduced. Ownership are evaluated by applying the EM algorithm in [22],[25] as:

$$\begin{aligned} O_{c,t,xy}^j &\propto m_{c,t,xy}^j N(\mathbf{v}_t^j; \boldsymbol{\mu}_{c,t}^j, \boldsymbol{\Sigma}_{c,t}^j) \\ O_{c,t,x}^j &\propto m_{c,t,x}^j \mathcal{N}\left(v_{t,x}^j; \mu_{c,t,x}^j, (\sigma_{c,t,x}^j)^2\right) \\ O_{c,t,y}^j &\propto m_{c,t,y}^j \mathcal{N}\left(v_{t,y}^j; \mu_{c,t,y}^j, (\sigma_{c,t,y}^j)^2\right) \end{aligned} \tag{13}$$

and $\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} O_{c,t,x}^j = 1$, $\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} O_{c,t,y}^j = 1$, $\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} O_{c,t,xy}^j = 1$, where $\mathcal{N}\left(v_{t,x}^j; \mu_{c,t,x}^j, (\sigma_{c,t,x}^j)^2\right)$ is the normal density function. The Ownerships are subsequently used for updating the mixing probabilities (parameter α is as previously defined, $\alpha = 1 - e^{-1/\tau}$):

$$\begin{aligned} m_{c,t+1,x}^j &= \alpha O_{c,t,x}^j + (1-\alpha) m_{c,t,x}^j \\ m_{c,t+1,y}^j &= \alpha O_{c,t,y}^j + (1-\alpha) m_{c,t,y}^j \\ m_{c,t+1,xy}^j &= \alpha O_{c,t,xy}^j + (1-\alpha) m_{c,t,xy}^j \end{aligned} \tag{14}$$

and $\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} m_{c,t,x}^j = 1$, $\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} m_{c,t,y}^j = 1$, $\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} m_{c,t,xy}^j = 1$.

We compute the new mean values and the new covariance matrices for each motion vector by utilizing the first and second order data moments. First order data moments are updated as:

$$\begin{aligned} M_{1,t+1,x}^j &= \alpha O_{\mathbb{S},t,x}^j v_{t,x}^j + (1-\alpha) M_{1,t,x}^j \\ M_{1,t+1,y}^j &= \alpha O_{\mathbb{S},t,y}^j v_{t,y}^j + (1-\alpha) M_{1,t,y}^j \\ M_{1,t+1,xy}^j &= \alpha O_{\mathbb{S},t,xy}^j v_{t,x}^j v_{t,y}^j + (1-\alpha) M_{1,t,xy}^j. \end{aligned} \tag{15}$$

Second order data moments are updated as:

$$\begin{aligned} M_{2,t+1,x}^j &= \alpha O_{\mathbb{S},t,x}^j (v_{t,x}^j)^2 + (1-\alpha) M_{2,t,x}^j \\ M_{2,t+1,y}^j &= \alpha O_{\mathbb{S},t,y}^j (v_{t,y}^j)^2 + (1-\alpha) M_{2,t,y}^j. \end{aligned} \tag{16}$$

The stable component is updated using the first order data moments:

$$\begin{aligned} s_{t+1,x}^j &= \mu_{\mathbb{S},t+1,x}^j = \tfrac{M_{1,t+1,x}^j}{m_{\mathbb{S},t+1,x}^j} \\ s_{t+1,y}^j &= \mu_{\mathbb{S},t+1,y}^j = \tfrac{M_{1,t+1,y}^j}{m_{\mathbb{S},t+1,y}^j}. \end{aligned} \tag{17}$$

The stable component new covariance matrices are evaluated as:

$$\begin{aligned} (\sigma_{\mathbb{S},t+1,x}^j)^2 &= \tfrac{M_{2,t+1,x}^j}{m_{\mathbb{S},t+1,x}^j} - (s_{t+1,x}^j)^2 \\ (\sigma_{\mathbb{S},t+1,y}^j)^2 &= \tfrac{M_{2,t+1,y}^j}{m_{\mathbb{S},t+1,y}^j} - (s_{t+1,y}^j)^2 \\ (\sigma_{\mathbb{S},t+1,xy}^j)^2 &= \tfrac{M_{1,t+1,xy}^j}{m_{\mathbb{S},t+1,xy}^j} - (s_{t+1,x}^j)(s_{t+1,y}^j). \end{aligned} \tag{18}$$

The wander component contains the current motion vectors, since it adapts as a two frame motion change model:

$$\begin{aligned} w_{t+1,x}^j &= \mu_{\mathbb{W},t+1,x}^j = v_{t,x}^j \\ w_{t+1,y}^j &= \mu_{\mathbb{W},t+1,y}^j = v_{t,y}^j. \end{aligned} \tag{19}$$

Covariance matrices for the wander and lost components are set equal to the estimated stable component covariance matrice $\hat{\boldsymbol{\Sigma}}_{\mathbb{W},t+1}^j = \hat{\boldsymbol{\Sigma}}_{\mathbb{L},t+1}^j = \hat{\boldsymbol{\Sigma}}_{\mathbb{S},t+1}^j$. Moreover, as it has been designed, component $\mathbb{L}_t$ remains constant by setting: $\hat{l}_{t+1,x}^j = \hat{l}_{1,x}^j = 0$ and $\hat{l}_{t+1,y}^j = \hat{l}_{1,y}^j = 0$.

E. Model Initialization

To initialize $OVFM_1$, the array $\mathbf{R} = [\mathbf{r}_x \quad \mathbf{r}_y]$ is considered, where $\mathbf{r}_x$ and $\mathbf{r}_y$ are $n \times 1$ vectors containing the motion vectors residuals in the x and y directions obtained from the first two frames of the video sequence. The stable, wander and lost components of the $OVFM_1$ model are then initialized by setting: $\hat{\mathbf{S}}_{1,x} = \hat{\mathbf{W}}_{1,x} = \mathbf{r}_x$,$\hat{\mathbf{S}}_{1,y} = \hat{\mathbf{W}}_{1,y} = \mathbf{r}_y$, $\hat{\mathbf{L}}_{1,x} = \hat{\mathbf{L}}_{1,y} = 0$.

Moreover, the covariance matrices and mixing probabilities for each component, as well as, the first and second order data moments are instantiated as follows: $\hat{m}^j_{c,1,x} = \hat{m}^j_{c,1,y} = \hat{m}^j_{c,1,xy} = \frac{1}{3}$,$\hat{\mathbf{\Sigma}}^j_{\mathbb{S},1} = \begin{bmatrix} (\hat{\sigma}^j_{\mathbb{S},1,x})^2 & (\hat{\sigma}^j_{\mathbb{S},1,xy})^2 \\ (\hat{\sigma}^j_{\mathbb{S},1,xy})^2 & (\hat{\sigma}^j_{\mathbb{S},1,y})^2 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.15 \\ 0.15 & 0.25 \end{bmatrix}$, $\hat{\mathbf{\Sigma}}^j_{\mathbb{W},1} = \hat{\mathbf{\Sigma}}^j_{\mathbb{L},1} = \hat{\mathbf{\Sigma}}^j_{\mathbb{S},1}$, $\hat{M}^j_{1,1,x} = \hat{s}^j_{1,x}\hat{m}^j_{\mathbb{S},1,x}$, $\hat{M}^j_{1,1,y} = \hat{s}^j_{1,y}\hat{m}^j_{\mathbb{S},1,y}$, $\hat{M}^j_{1,1,xy} = \left((\hat{\sigma}^j_{\mathbb{S},1,xy})^2 + \hat{s}^j_{1,x}\hat{s}^j_{1,y}\right)\hat{m}^j_{\mathbb{S},1,xy}$, $\hat{M}^j_{2,1,x} = \left((\hat{\sigma}^j_{\mathbb{S},1,x})^2 + (\hat{s}^j_{1,x})^2\right)\hat{m}^j_{\mathbb{S},1,x}$, $\hat{M}^j_{2,1,y} = \left((\hat{\sigma}^j_{\mathbb{S},1,y})^2 + (\hat{s}^j_{1,y})^2\right)\hat{m}^j_{\mathbb{S},1,y}$, where $c \in \{\mathbb{S}, \mathbb{W}, \mathbb{L}\}$.

F. State Transition

In various particle filter applications, the quantity of the system disturbance during state transition plays a crucial role in the state estimation process. By measuring the system disturbance during the previous states, we can infer the expected system disturbance at a future state. This approach creates the necessity for an adaptive state transition model. In our approach, as will be discussed below, we have incorporated the system disturbance momentum in order to regulate the applied noise variance and to resize the generated particle filters set. In [26], system disturbance is measured as the sum of the absolute difference between the states corresponding to successive video frames. This parameter is associated with the decision that is acquired in order to switch between a deterministic and a stochastic search method that is used for each particle. Moreover in [24], the tracked object velocity is measured as the shift in the state vector between two consecutive frames and is computed using a first order Taylor series expansion around a current state estimate. The computed velocity usually indicates the minimization direction of the difference between the compared image patches and is exploited in order to further stabilize the tracker by fine tuning around the state estimate with the highest likelihood.

In this approach, the motion vector field $\mathbf{Y}_t$ is available (e.g., from block matching) when a new frame is processed, in contrast with the previously presented approaches, where the exact block position, inside the current video frame t, could only be approximated using the state estimate $\hat{\boldsymbol{\theta}}_t$. As a result, we can evaluate our estimation error e_t, by measuring the distance between the estimated motion vector field $\hat{\mathbf{Y}}_t$, that the $OVFM_t$ model contains and the actual motion vector field we obtain, as:

$$\begin{aligned} e_t &= 1 - P(\mathbf{Y}_t|\hat{\boldsymbol{\theta}}_t) \\ &= 1 - \prod_{j=1}^{n}\left\{\sum_{c=\mathbb{S},\mathbb{W},\mathbb{L}} m^j_{c,t,xy} N\left(\mathbf{v}^j_t; \boldsymbol{\mu}^j_{c,t}, \mathbf{\Sigma}^j_{c,t}\right)\right\}, \end{aligned} \tag{20}$$

where the vector $\mathbf{Y}_t = [\mathbf{v}^1_t \dots \mathbf{v}^n_t]$ contains the motion vector field.

We exploit the computed estimation error e_t, in order to dynamically adjust not only the applied noise variance, but also the population of particle filters that will be generated in the following estimation process. The complete state transition method is summarized by the equation:

$$\hat{\boldsymbol{\theta}}_t = \boldsymbol{\theta}_{t-1} + \mathbf{U}_{t-1}, \tag{21}$$

where $\mathbf{U}_{t-1}$ is the applied noise. In order to reduce the computational load and enhance the robustness of the proposed algorithm, we have adapted strategies that are described in the following two subsections.

G. Adaptive Noise - Adaptive Number of Particles

In the applied particle filter framework, the introduced noise variance and the generated particle filters population size, severely affect the accuracy of the camera motion parameter estimation. Concisely, it is noted that the size of the search space that is covered in each search iteration is proportional to the variance of the applied noise. Larger noise variance enables searches in broader regions of the state space, thus allowing the model to adapt to severe changes in the motion parameter state. On the other hand, smaller noise variance enables the model to fine tune around a persistent motion parameter state. In addition, the accuracy of the estimation is proportional to the number of the used particle filters. More particle filters offer greater coverage of possible motion states while demanding greater computational effort.

We exploit these characteristics in order to find the optimal trade off between the estimation accuracy and the required computational effort. Our intention is to dynamically adjust the noise variance and the number of processed particles, so as to generate fewer number of particles with small noise variance, when small changes in the camera motion are required. When large jumps in the motion state space need to be covered, we adjust our settings so as to process a larger number of particle filters with larger noise variance.

We evaluate the accuracy of our previous prediction by computing the estimation error e_t. Subsequently, the number of the processed particle filters A_t and the applied noise variance σ_t, are adjusted for the following prediction step proportionally to the estimation error, according to the formulae:

$$\begin{aligned} A_t &= \min(\tfrac{A_{\min}}{e_t}, A_{\max}) \\ \sigma_t &= \min(\tfrac{\sigma_{\min}}{e_t}, \sigma_{\max}), \end{aligned} \tag{22}$$

where both the population of the generated particles, as well as, the noise variance, are bounded in order to ensure computational efficiency, algorithm robustness and optimal performance. In our experiments the number of particles have been between 150 and 300.

H. Robust Parameter Estimation

Data outliers are common in motion vector fields and in order to further stabilize the system in such settings an additional data pre processing step has been applied that enables the system not only to statistically identify data outliers but

also to reform those motion vectors that have been obviously assigned invalid values.

To address this problem, a 3×3 spatial median filter is initially applied to the motion vector field x,y components, thus reducing motion vector field outliers appearing in homogeneous video frame regions. Moreover, we use an iterative bivariate Winsorization transform [27], [28] which provides a mechanism for data outlier detection and rectification. For each motion vector $\mathbf{v}_t^i = [v_{t,x}^i \quad v_{t,y}^i]^T$, the Mahalanobis distance $D(\mathbf{v}_t^i)$ is computed based on an initial bivariate covariance matrix $\hat{\boldsymbol{\Sigma}}_t$ and mean value $\hat{\boldsymbol{\mu}}_t$ estimates, according to the formula:

$$D(\mathbf{v}_t^i) = (\mathbf{v}_t^i - \hat{\boldsymbol{\mu}}_t)^T \hat{\boldsymbol{\Sigma}}_t{}^{-1} (\mathbf{v}_t^i - \hat{\boldsymbol{\mu}}_t) \tag{23}$$

where $\hat{\boldsymbol{\mu}}_t = [\hat{\mu}_{t,x} \hat{\mu}_{t,y}]^T$ and $\hat{\boldsymbol{\Sigma}}_t = diag\{\frac{\delta_{t,x}}{0.6745}, \frac{\delta_{t,y}}{0.6745}\}$ where $\hat{\mu}_{t,x}$, $\hat{\mu}_{t,y}$ are the robust mean values, $\delta_{t,x}$ and $\delta_{t,y}$ are the adjusted mean absolute deviations computed from the motion vector components along the x and y axes, respectively. We treat a motion vector as an outlier when its Mahalanobis distance $D(\mathbf{v}_t^i) > T$, where T is a positive constant. Based on experimental evidence we choose $T = 5.99$ which gives 95% efficiency at the $\mathcal{X}_2^2$ distribution. Rectification of detected data outliers is performed by truncating such motion vectors to the border of a two-dimensional ellipse which contains the majority of the motion vectors, by using the bivariate transformation:

$$\tilde{\mathbf{v}}_t^i = [\tilde{v}_{t,x}^i \ \tilde{v}_{t,y}^i]^T = \hat{\boldsymbol{\mu}}_t + \min\Big(\sqrt{\frac{T}{D(\mathbf{v}_t^i)}}, 1\Big)\big(\mathbf{v}_t^i - \hat{\boldsymbol{\mu}}_t\big). \tag{24}$$

The process is recursively executed until no more motion vector field outliers are detected. In a final data pre-processing step, a whitening transform is applied to the motion vector field. The data set is transformed so that the motion vectors have zero mean value and their covariance matrix $\hat{\boldsymbol{\Sigma}}_t$ is equal to the identity matrix.

I. Conformal Affine Transform

The restricted 2-D affine transformation model includes four affine parameters, thus constituting the more appropriate parametric model in describing the camera motion, if we neglect the introduced lens distortion. According to this transformation only conformal scaling and rotation along the x and y axes video frame deformation is performed, due to camera motion. The 2-D affine transformation of the i-th block center displaced from position (x_i, y_i) to (x_i', y_i') according to this model is given by:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha cos\phi & -sin\phi & T_x \\ sin\phi & \alpha cos\phi & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{25}$$

α, ϕ, T_x, T_y correspond to scaling by a factor α, rotation by ϕ degrees and translation by T_x and T_y pixels along the direction of x and y axes, respectively.

We generate conformal scaling and rotational potential future states, when the state estimates set is populated, by regulating the applied noise in each particle equivalently for the respective affine transform parameters.

IV. Experimental Results

We evaluate the efficiency of the proposed method through extensive experimental testing. The testing dataset comprises of 30 edited outdoor video streams, including in total $26,245$ frames, while the motion vector fields have been obtained by applying the block matching algorithm. The dataset includes all patterns of distinct camera motion (zoom in, zoom out, pan, tilt and rotation) and also combinations of them. Moreover, since the presented algorithm not only identifies the performed camera motion pattern but also measures the motion parameters, we have included in our test collection video streams that contain sequential frame regions, where the camera moves according to a specific pattern but at a variable pace.

A. Camera Motion Pattern Classification

The acquired after each prediction process, affine transform parameters, are used to infer the type of the performed camera motion. However, due to the fact that we generate our solutions set in each prediction step by adding random noise, our method, as every stochastic approach, moves around the optimum solution. This fact introduces some error in the classification of the performed camera motion, when the affine transform coefficients vary around critical boundaries, in terms of camera motion interpretation. To alleviate this error in the camera motion characterization, we assume that any camera motion, in order to be classified as of a specific pattern, should have a minimum duration of five consecutive frames, otherwise, it is absorbed by preceding or succeeding dominant camera motions. Moreover, in order to further stabilize our camera motion detection method, we filter the obtained affine transform coefficients set by applying a temporal median filter having window size 3.

We interpret the affine transform coefficients contained in the state vector $\boldsymbol{\theta}_t$ as follows:

- If $\grave{m}_1 = \frac{m_1}{cos\phi} > 1$, then the detected camera motion is classified as zoom in.
- If $\grave{m}_1 = \frac{m_1}{cos\phi} < 1$, then the detected displacement is characterized as zoom out.
- If $m_2 < 0$, the camera rotates in a clockwise manner.
- If $m_2 > 0$, the camera rotates in an anti-clockwise manner.
- Parameters m_3 and m_4 define pan and tilt along the direction of x and y axes, respectively.

We provide experimental results obtained by applying the proposed method in representative video sequences for each camera motion pattern. The variation of the affine coefficients describing the camera motion at each video frame it is presented at the accompanying graphs. Moreover, at key moments, when the camera motion pattern alters, the respective video frames are provided for visual confirmation of the obtained results.

In Fig. 1, the results that are obtained by applying the proposed method in a video sequence comprised of 992 frames where the camera performs pan and tilt, are presented. The variation of the affine coefficients responsible for translation according to x and y axes, it is sketched in this graph, as the test video evolves over time. As shown, the camera pans to

the right during the frame intervals $1-107$ and $250-352$, while it pans to the left at the frame intervals $108-249$ and $805-918$. Moreover, during the video frames $353-437$ and $624-681$ the camera tilts up, while during the temporal intervals $438-560$ and $919-992$ the camera tilts down. Finally, during the interval $561-623$ camera stands still, while from frame 718 and until frame 804 camera moves diagonally up and to the right.

The lower graph in Fig. 1 presents the variation of the mixture probabilities that regulate each component's contribution to the observation likelihood derived from the same video stream. As it is observed the stable component's membership initially declines, as expected, since the model has not created an accurate motion representation during that period. On the other hand, the wander component adapts faster than the stable, as it has been designed, and as a result its contribution during the same temporal interval increases. In general, the mixing probability of the stable component reaches its highest value, at the exact moment the camera completes a distinct motion pattern, since at that time the stable component has the optimum smoothed camera motion representation. On the other hand, at the same moment the wander component's membership is assigned its lowest value.

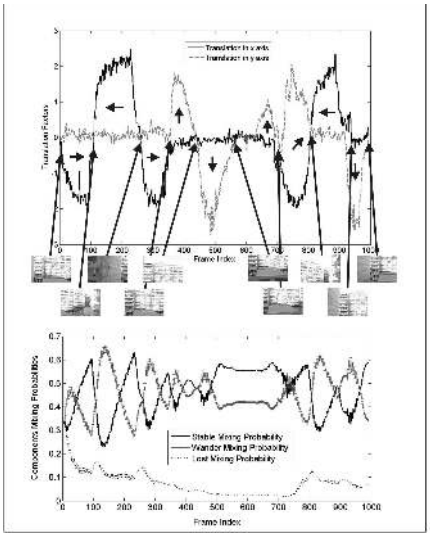

Fig. 1. Variation of translation factors m_3 and m_4 according to x and y axes, respectively. The lower graph shows the variation of the components mixing probability as the video stream evolves. Each model component identifies a different type of camera motion.

The next examined test video sequence contains 237 frames, in which the camera zooms in and out, while there are sequential video frame temporal regions where camera remains still. Fig. 2 depicts the variation of the obtained scale factor $\dot{m}_1$. The proposed method successfully identified and classified the performed camera motion in three different patterns. During the frame interval $37-114$, camera motion has been classified as zoom in, while for the frame interval $152-232$, it has been characterized as zoom out. There are three groups of sequential frame regions ($1-36$, $115-152$ and $232-237$) where the proposed algorithm has not detected any significant camera motion and these periods have been characterized as still ones.

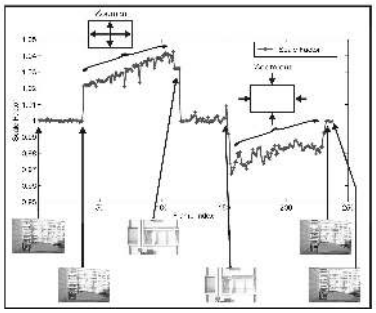

Fig. 2. Variation of the scale factor when the camera performs zooming in and out. At key frames when camera changes its motion pattern video frames are provided.

Fig. 3 presents a camera motion case for a video of 695 frames, where the camera rotates in a clockwise and in an anti-clockwise manner, while there are 9 video frame intervals, where the camera remains stationary. These labelled video frame groups either have been successfully detected and characterized or they have been absorbed by preceding or succeeding dominant camera motions. It should be noted that according to the conformal affine transformation model, the rotation coefficient corresponds to $m_2 = sin\phi$. The camera motion has been identified and classified as follows: rotation in a clockwise manner inside the video frame intervals $12-197$ (region 2 has been absorbed), $565-615$ and $625-660$ and rotation in an anti-clockwise manner in the video frame interval $213-564$ (except from the labelled regions 4, 5, 6 and 7 which have not been absorbed). Finally, the performed camera motion pattern has been characterized as stationary inside the labelled video frame intervals 1,3,4,5,6,7,8 and 9.

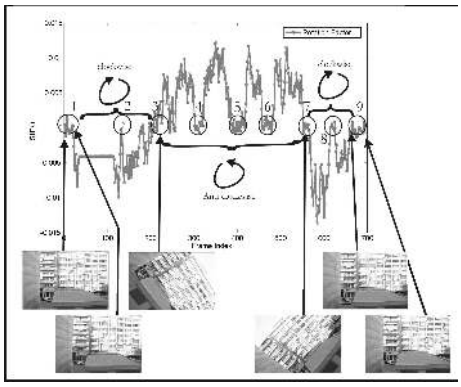

Fig. 3. Variation of $sin\phi$ which declares the affine coefficient responsible for rotation. Nine regions are distinguished in which the camera remains stationary.

B. Camera Motion Modelling Accuracy

In each prediction step, the system state vector defined in (1), is adjusted so as to approximate the affine transform coefficients that better fit the motion representation that $OVFM_t$

contains. On the other hand, the predicted future state determines a motion vector field which could be obtained by repositioning the video frame block centers, as determined by the affine transformation and compute each block displacement with respect to its previous position. Therefore, since the real motion vector field is available when a new frame is processed, the accuracy of the last prediction could be assessed by computing the Mean Square Error (MSE) between the estimated and the real motion vector fields:

$$MSE_t = \frac{1}{n}\sum_{i=1}^{n}\left(\left(\hat{v}_{t,x}^i - v_{t,x}^i\right)^2 + \left(\hat{v}_{t,y}^i - v_{t,y}^i\right)^2\right), \quad (26)$$

where $\hat{\mathbf{v}}_t^i = [\hat{v}_{t,x}^i \quad \hat{v}_{t,y}^i]$ and $\mathbf{v}_t^i = [v_{t,x}^i \quad v_{t,y}^i]$ correspond to the i-th estimated and real motion vector at time t, respectively.

The proposed method has been applied in a video sequence containing 186 frames, where the camera zooms in with variable pace, except from a temporal interval between frames $181 - 186$, where it remains still. Fig. 4 presents the MSE produced by the LS solution and the proposed method. As depicted, while the camera zooms in at a growing pace from the beginning of the video sequence until frame 180 (during this interval the scaling factor has tripled), the increase in the scale factor value is also followed by an increase in the generated MSE. The radical drop in the MSE at frame 181 occurs since the camera changes its motion pattern and remains still. As depicted, the proposed method constantly generates lower error compared with the LS solution. During the complete video sequence the average generated MSE by the proposed method is $\overline{MSE} = 0.588$, while the LS solution produces on average almost a triple value since $\overline{MSE} = 1.6744$.

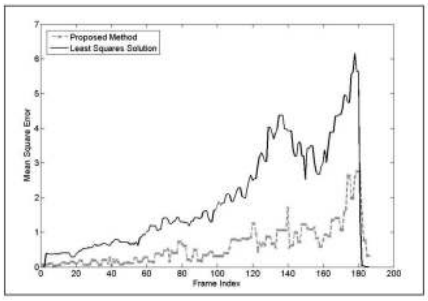

Fig. 4. Computed MSE error produced by the LS solution and the proposed method.

Fig. 5 shows the MSE obtained for the video sequence presented in Fig. 3. The nine video frame temporal intervals, in which the camera remains still, are distinctive, since the majority of the contained motion vectors are equal to zero and as a result, the generated error is minimal. The proposed method clearly outperforms the LS solution, since the computed MSE is constantly lower than 1.0. Moreover, the average MSE introduced by the LS solution is $\overline{MSE} = 0.65847$, while the corresponding average MSE for the proposed method is $\overline{MSE} = 0.1977$.

In Fig. 6, average MSE generated by both solutions and computed over the complete test dataset is presented. As can be seen, the proposed method clearly generates smaller $\overline{MSE}$ values, independently of the performed camera motion pattern.

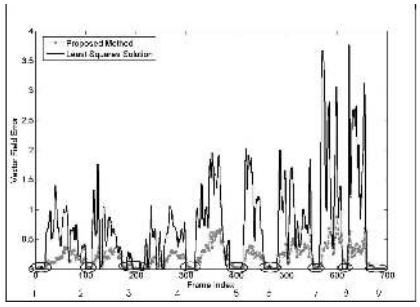

Fig. 5. Comparison of the produced MSE between the proposed method and the LS solution. In this video, the camera rotates while there exist nine sequential frame regions in which it remains still.

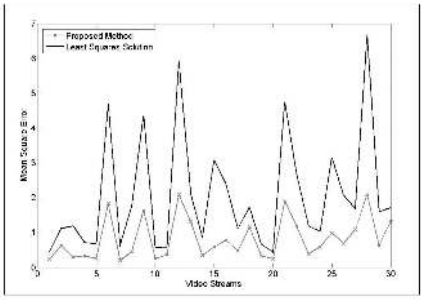

Fig. 6. $\overline{MSE}$ computed for each video in the test dataset.

C. Compressed Video

In order to reduce the processing time, we have applied the proposed method directly on compressed MPEG video streams without performing full frame decompression in advance. MPEG video streams are composed of an hierarchically organized structure [29], [30] consisting of: sequences, Group Of Pictures (GOP), pictures, slices, macroblocks and blocks. A GOP consists of three different types of pictures: the I-frames which are coded pictures using only information present in the picture itself, the predicted pictures (P-frames) coded with respect to the nearest previous I- or P-frame and the bidirectionally predicted pictures (B- frames) coded using both a past and a future I- or P-frame as a reference.

Since I-frames are intracoded and B-frames are coded bidirectionally, we can neglect them and apply our method directly to the resulting P-frames exploiting the contained motion vectors. As a result of this approach, significant computational gain is observed, since we essentially sub sample over time. On the other hand, we expect an increase in the estimation error, especially in video sequences in which rapid changes in camera motion occur frequently. Fig. 7 presents a comparison of the MSE obtained by applying the proposed method either in the complete video stream or only in P-frames. As expected, the difference in the MSE is smaller when camera moves smoothly, since there are no radical

variations in the motion vector fields across P-frames, as for instance, during the frame intervals $1-38$ and $120-177$. On the other hand, we observe severe performance difference between the two approaches, when the camera moves faster. Moreover, the average MSE computed from the uncompressed video stream is $\overline{MSE} = 0.20496$ while for the compressed MPEG video is $\overline{MSE} = 0.32213$.

Furthermore, the time required to perform camera motion estimation has been evaluated. Experiments have been conducted on an Intel Pentium 4 processor, running at 3.0 Ghz and using 1 GB of RAM. The under examination video consists of 237 frames, sized 360×240 pixels, where 61 of them are P-frames. Since for the MPEG coder each block is of dimension 8×8 pixels, we have applied the same settings in the block matching algorithm in order to obtain equally sized motion vector fields. The system requires 0.144 seconds to process one video frame or, equivalently, it processes 6.93 motion vector fields per second, which is a prohibitive amount of time for real time systems. However, in compressed MPEG videos of NTSC quality, coded in a rate of 30 frames per second, there are available 8 P-frames per second, which means that such MPEG video streams can be processed almost in real time.

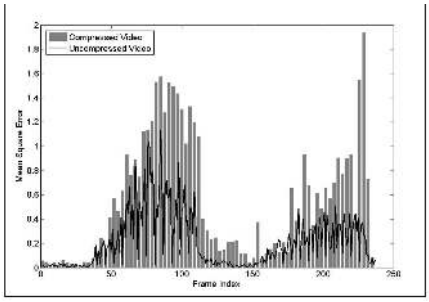

Fig. 7. MSE computed for the same compressed MPEG and uncompressed video stream.

D. Incorporation of Camera Motion Estimation with Moving Object Tracking.

The effectiveness of incorporating the proposed method within an object tracker in order to separate the camera motion from the tracked object motion, has been also investigated. We have embedded our algorithm within the object tracker proposed in [24], which also predicts the future position of the tracked object. We have applied the new schema in a video sequence in which, while the camera zooms in, the tracked object moves along the y axis. Moreover, since tracking is performed in each frame of the video sequence, we obtain the motion vector fields using a block matching algorithm. Firstly, the proposed method determines the affine transformation describing the camera motion and then launches the object tracker. Both state vectors denote affine transformations therefore, the generated future state estimates by the object tracker $\{\boldsymbol{\theta}_{o,t}^{i}\}_{i=1}^{K}$, determining the position of the image region of interest inside the video frame, are transformed prior their evaluation as:$\{\grave{\boldsymbol{\theta}}_{o,t}^{i}\}_{i=1}^{K} = \boldsymbol{\theta}_t\{\boldsymbol{\theta}_{o,t}^{i}\}_{i=1}^{K}$. In Fig. 8, screen shots of the tracking process with and without camera motion incorporation are provided. Both tracking processes have been initialized to track exactly the same regions. Fig. 9a presents the tracking results where camera motion estimation has not been incorporated. Fig. 9b presents the obtained results where camera motion prevention has been included, while Fig. 9c shows the ground truth for the respective video frames. Notice that the bounding box has been scaled up during the video sequence since the detected camera motion is zoom in. We have quantitatively measured the performance of the new schema by comparing the spatial overlap amount between the highlighted by the object tracker region and the ground truth data with and without considering the camera motion. The obtained measurements for each video frame have been sketched in a graph, shown in Fig. 9. In the examined video sequence the mean spatial overlap with respect to the ground truth data has been increased by 30.4%.

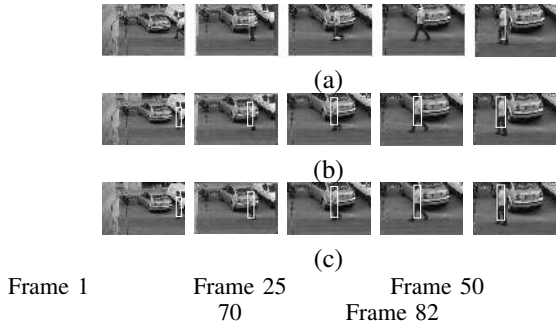

Fig. 8. a) The tracking results where camera motion detection has not been included. b) Tracking results of the new schema. c) Ground truth.

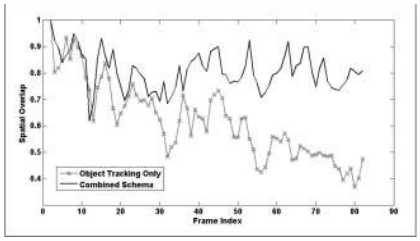

Fig. 9. Spatial overlap amount between tracks obtained with and without considering the camera motion.

V. Conclusion

Our main aim in this work is to determine accurately the motion parameters of the performed camera movement and not only to identify the performed 2D motion pattern. To do so, a novel camera motion estimation method based on using the motion vector field has been presented in this paper. The features that distinguish our method from other proposed camera motion estimation techniques are: 1) the integration of a novel stochastic vector field model, 2) the incorporation of the vector field model inside a particle filters framework where an online EM algorithm for model parameters update enables the method to estimate the future camera movement and 3) the ability to detect, characterize and estimate the

performed camera motion pattern. Motivated by the fact that camera motion could be temporarily characterized by rapidly varying, slowly varying and stationary movement patterns, we have designed the proposed model, that easily adapts to camera motion types having possibly variable pace. Extensive experimental results have verified that the proposed method not only successfully characterizes the detected camera motion pattern but also predicts the subsequent performed camera motion with minimal error.

REFERENCES

[1] W. Lie and W. Hsiao, "Content-based video retrieval based on object motion trajectory," in *IEEE Workshop on Multimedia Signal Processing*, December 2002, pp. 237–240.

[2] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank, "Semantic-based surveillance video retrieval," *IEEE Transactions on Image Processing*, vol. 16, no. 4, pp. 1168 – 1181, April 2007.

[3] Y. Jianfeng and L. Zhanhuai, "Modeling of moving objects and querying videos by trajectories," in *MMM '04: Proceedings of the 10th International Multimedia Modelling Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 373–380.

[4] Y.-P. Tan, D. D. Saur, S. R. Kulkarni, and P. J. Ramadge, "Rapid estimation of camera motion from compressed video with application to video annotation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 133–145, 2000.

[5] L.-Y. Duan, J. S. Jin, Q. Tian, and C.-S. Xu, "Nonparametric motion characterization for robust classification of camera motion patterns," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 323–340, 2006.

[6] X. Zhu, A. K. Elmagarmid, X. Xue, L. Wu, and A. C. Catlin, "Insightvideo: Toward hierarchical video content organization for efficient browsing, summarization and retrieval," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 648–666, 2005.

[7] T. Lertrusdachakul, T. Aoki, and H. Yasuda, "Camera motion characterization through image feature analysis," in *ICCIMA'05*, 16-18 Aug. 2005, pp. 186 – 190.

[8] H. Yi, D. Rajan, and L.-T. Chia, "Automatic generation of mpeg-7 compliant xml document for motion trajectory descriptor in sports video," *Multimedia Tools Appl.*, vol. 26, no. 2, pp. 191–206, 2005.

[9] M. Ben-Ezra and S. K. Nayar, "Motion-based motion deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 689–698, 2004.

[10] C. Cotsaces, N. Nikolaidis, and I. Pitas, "Video shot detection and condensed representation. a review," *Signal Processing Magazine, IEEE*, vol. 23, no. 2, pp. 28–37, March 2006.

[11] H. J. Zhang, C. Y. Low, S. W. Smoliar, and J. H. Wu, "Video parsing, retrieval and browsing: an integrated and content-based solution," in *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 1995, pp. 15–24.

[12] Y. F. Ma, L. Lu, H. J. Zhang, and M. Li, "A user attention model for video summarization," in *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 2002, pp. 533–542.

[13] M. Irani and P. Anandan, "Video indexing based on mosaic representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 86, no. 5, pp. 905–921, May 1998.

[14] Z. Duric and A. Rosenfeld, "Stabilization of image sequences," College Park, MD, USA, Tech. Rep., 1995.

[15] Y. S. Yao and R. Chellappa, "Electronic stabilization and feature tracking in long image sequences," College Park, MD, USA, Tech. Rep., 1995.

[16] H. Jozawa, K. Kamikura, A. Sagata, H. Kotera, and H. Watanabe, "Two-stage motion compensation using adaptive global MC and local affine MC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 75–85, 1997.

[17] S. F. Wu and J. Kittler, "A differential method for simultaneously estimation of rotation, change of scale and translation," *Signal Process. Image Commun.*, vol. 2, no. 1, pp. 69–80, 1990.

[18] J.-G. Kim, H.-S. Chang, J. Kim, and H.-M. Kim, "Efficient camera motion characterization for MPEG video indexing," in *ICME '00*, vol. 2, 30 July-2 Aug. 2000, pp. 1171 – 1174.

[19] F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 497–501, 2000.

[20] J. H. Kotecha and P. M. Djuric, "Gaussian particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2592–2601, October 2003.

[21] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

[22] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1296–1311, 2003.

[23] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian bayesian state estimation," in *IEE Proc.-F*, vol. 140, no. 2, April 1993, pp. 107–113.

[24] S. K. Zhou, R. Chellappa, and B. Moghaddam, "Visual tracking and recognition using appearance-adaptive models in particle filters." *IEEE Transactions on Image Processing*, vol. 13, no. 11, pp. 1491–1506, 2004.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[26] J. Sullivan and J. Rittscher, "Guiding random particles by deterministic search," in *ICCV*, 2001, pp. 323–330.

[27] J. Chilson, R. Ng, A. Wagner, and R. Zamar, "Parallel computation of high-dimensional robust correlation and covariance matrices," *Algorithmica*, vol. 45, no. 3, pp. 403–431, 2006.

[28] P. Huber, *Robust Statistics*. Wiley, New York, 1981.

[29] A. M. Tekalp, *Digital video processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[30] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.