Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

The Markovian Jump Contour Tracker [⋆]

Albino A. A. Cordeiro Jr. [*] **Marcelo D. Fragoso** [*]
Nicolas D. Georganas [**] **Jauvane C. de Oliveira** [*]

[*] *Laboratorio Nacional de Computacao Cientifica, Brazil*
(e-mail: acordeiro@discover.uottawa.ca)
[**] *University of Ottawa, Canada*

Abstract: In this paper a novel contour tracking algorithm based on a *Markovian jump* extension of the Kalman filter is presented, we call it the Markovian jump tracker (MJT). Markovian jump linear systems (MJLS) are suitable to model physical systems that behaves linearly but suffers abrupt changes in the dynamics from time to time (accordingly to a Markov chain process) with applications ranging from aircraft control systems to macroeconomics simulation. The tracking method we devised uses Markovian jump filter technology to profit from available multiple linear models -to achieve overall tracker stability- each roughly describing a specific type of expected motion, for example: rotation, translation, cyclic motion, and so on. We verify the method's efficacy in preliminary experiments and compare aspects of its performance with other popular contour tracking algorithm.

Keywords: Switched discrete and hybrid systems, Linear Parametrically Varying (LPV) Methodologies; Estimation and Filtering.

1. INTRODUCTION

The contour is one of the most descriptive visual feature of an object, thus, an effective contour tracking algorithm features out as a very desirable tool for many computer vision applications. Contour tracking in natural video is a rather challenging task, for many factors concur to distract the tracker and to degrade performance such as, for instance: change in illumination, weak color contrast, intrinsic image noise, and other objects in the scene that may mimic some of the target's features.

Often, a single linear dynamic model is used by the tracker to estimate motion for the whole video, although, throughout the video the actual target motion dynamic (or operation mode) usually varies significantly. The ideal solution would consider this dynamic variation during motion, such that for each time step the appropriate dynamic model would be utilized by the tracker.

In this work we deal with the case where multiple dynamic models are available, each roughly describing a certain kind of motion that is expected to be present in the target videos, for example, a model for rotation, another for in-plane translation, one more for scale variation, and possibly models for combinations of these types of motion. In practice, these models are *learned* from available data. We assume also that the transitions between motion modes occur in such small time intervals that we may consider them as *abrupt* changes in the dynamics.

This perhaps restrictive case, nonetheless, cover important real-life applications such as, for example, hand tracking for gesture-based human-computer interfaces (see Tomasi et al. (2003) and Athitsos and Sclaroff (2002)) which was the underlying motivation of this work.

In the kernel of our contour tracker is an optimal Markovian jump filter, that is presented in Costa et al. (2005) combined with a contour modeling and measurement framework found in Blake and Isard (1998). The resulting algorithm, aside other interesting characteristics, is simple to understand -since it is a generalization of the Kalman filter- and it's also easy to implement, notwithstanding, it presents compelling empirical results.

1.1 Related Work

Multiple models methods for visual tracking has received significant attention in recent years and has been perceived as a strong tool for increasing robustness of traditional object tracking algorithms. These methods are also called hybrid methods in the literature, that is because an usual form to describe the problem is to utilize hybrid state vectors, such that, one of the coordinates evolves discretely and indicates which dynamic model is active at each time step. Important practical and theoretical results for hybrid systems have already been published (for example Doucet et al. (2000)).

In Nascimento and Marques (2004) the authors propose a multiple model tracker (and a model identification method) for improving robustness under cluttered background conditions. In this case a joint probability data association filter was devised.

On the other hand, in Isard and Blake (1998b), the authors propose a particle filter contour tracker adjusted to accept multiple models, a joint probability density function was conceived to comprise both the state and the model label in order to improve robustness of a hand-contour tracking

[⋆] This work was supported in part by CNPq under Grants 470527/2007-2 and 301740/2007-0 and by FAPERJ under Grant E-26/100579/2007.

system for gesture recognition. Other important filters and adaptations of the particle filter for multiple model systems could arguably substitute the filter used in that work, such as Blom and Bar-Shalom (1988) and Blom and Bloem (2004). These ones however were not yet tested in the contour tracking context.

One upfront difference between the other proposals mentioned in this section and our algorithm is that the later doesn't attempt to measure nor estimate the jump parameter directly. This is neither a positive nor a negative characteristic in general, but will be a useful one or not accordingly to the underlying application.

1.2 Paper Organization

In section 2, the basic elements of the problem are set forth and the problem itself is defined. In section 3, the Markovian Jump framework is introduced. In section 4, the tracker design is described and some aspects of the implementation are discussed. In section 5, some preliminary experiments on hand tracking are reported. In section 6, we conclude the paper and propose paths for future research.

2. PROBLEM FORMULATION

In the early 1990's, Andrew Blake and collaborators developed a very successful contour tracking framework in a series of papers, they based their work on B-splines for contour representation (see Blake and Isard (1998) and references therein). The developments we present here builds upon their work.

B-splines curves are parametric curves with many interesting properties: there are available well tested algorithms for point set interpolation -or approximation- with B-splines and, since B-splines curves are polynomial by parts, computer representation, manipulation and storage is, thus, greatly simplified, for all those operations reduce to manipulations with small vectors.

Basically, contours are described as linear combinations of elements of a set of spline functions previously defined, namely *spline basis*. The contour curve $\mathbf{r}_k$ in time step k can be described as matrix product:

$$\mathbf{r}_k(s) = U(s)^T \mathbf{Q}_k \ , 0 \leq s \leq L \tag{1}$$

where each $s \in [0, L]$ determines a point in $\mathbf{r}_k$, L is said to be the number of curve *spans* [1] , the column vector

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{Q}_k^x \\ \mathbf{Q}_k^y \end{pmatrix} \tag{2}$$

with $\mathbf{Q}_k^x, \mathbf{Q}_k^y \in \Re^{N_Q}$ is called the *control vector* [2] , and

$$U(s) = \begin{pmatrix} \mathbf{B}^T(s) & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^T(s) \end{pmatrix}$$

with $\mathbf{0} = (0, 0, ..., 0)^T \in \Re^{N_B}$ is a $2 \times 2N_Q$ matrix. For its turn, $\mathbf{B}(s)$ is called the spline basis vector and

$$\mathbf{B}^T(s) = (B_0(s), B_1(s), ..., B_{N_B-1}(s)) \tag{3}$$

where $B_j(s)$, $j = 0, ..., N_B - 1$, are the spline basis functions.

[1] L is a parameter of the spline formulation specifically. In our experiments L was set to 50.

[2] The control vector actually defines the curve shape at time step k.

The matrix $U(s)$ may be computed off-line for a fine discretization of the parameter space $[0, L]$, so that, retrieving points on the curve becomes the trivial task of calculating (1) given a control vector $\mathbf{Q}_k$. In practice N_Q is previously fixed (≈ 200).

We assume, henceforth, rigid body motion and that the distance camera-target is such that perspective effects may be ignored. Therefore, given the contour description for $k = 0$, $\mathbf{Q}_0$, every other subsequent contour disposition, $\mathbf{Q}_k$, is a result of an affine transformation of $\mathbf{Q}_0$, that is

$$\mathbf{Q}_k = W_0 \mathbf{X}_k + \mathbf{Q}_0 \tag{4}$$

where $\mathbf{X}_k \in \Re^{N_X}$ is the configuration vector for time-step k and

$$W_0 = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{Q}_0^x & \mathbf{0} & \mathbf{0} & \mathbf{Q}_0^y \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{Q}_0^y & \mathbf{Q}_0^x & \mathbf{0} \end{pmatrix} \tag{5}$$

with $\mathbf{1} = (1, 1, ..., 1)^T \in \Re^{N_B}$ defines all acceptable transformations of $\mathbf{Q}_0$, in fact, its column vectors form a basis for the space of affine transformations of $\mathbf{Q}_0$. The subset of contours $\mathbf{Q}_k$ for which (4) holds is said to be the *configuration* (or *shape*) *space* of $\mathbf{Q}_0$.

We assume also that we have some *a priori* knowledge of the dynamics in the form of a second-order autoregressive dynamic equation:

$$\mathbf{X}_k = A^2 \mathbf{X}_{k-2} + A^1 \mathbf{X}_{k-1} + B^0 \omega_k \tag{6}$$

where $A1$, A^2 and B^0 are $N_X \times N_X$ matrices, and ω_k is a vector of Gaussian random variables with zero mean and variance equals to 1.

However, defining the state vector

$$\mathbf{x}_k = \begin{pmatrix} \mathbf{X}_{k-1} \\ \mathbf{X}_k \end{pmatrix} \tag{7}$$

we're able to rewrite (6) as

$$\mathbf{x}_k = A \mathbf{x}_{k-1} + B \omega_k \tag{8}$$

where

$$A = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ A^2 & A^1 \end{pmatrix} \text{ and } B = \begin{pmatrix} \mathbf{0} \\ B^0 \end{pmatrix} \tag{9}$$

The state vector $\mathbf{x}_k$ is measured under noise, for which an observation model is assigned

$$\mathbf{y}_k = L \mathbf{x}_k + H \eta_k \tag{10}$$

where $\mathbf{y}_k$ is the observation or measurement, L the *observation coefficient* matrix, H the observation noise covariance, and η_k is defined in the same way as ω_k.

In this context we define the contour tracking problem as the problem of *recursive estimation* of the state vector $\mathbf{x}_k$, to be carried out over a n-dimensional *state space* [3] , given the dynamic model (8), the observation model (10), and the measurement process $\{\mathbf{y}_k\}$.

3. MARKOVIAN JUMP LINEAR SYSTEMS

MJLS describes processes subject to uncertain changes in their dynamics, where the variations caused by these changes significantly alter the dynamic behavior of the system. More specifically, a MJLS is characterized by a finite set of discrete-time linear systems with modal

[3] Note that $n \ll 2N_Q$, in fact, $n = 2N_X$ where $N_X = 6$ for the configuration space we're working with.

transition given by a Markov chain, that is, we consider systems of the form:

$$\mathcal{G} = \begin{cases} \mathbf{x}_{k+1} = A_{\theta_k}\mathbf{x}_k + B_{\theta_k}\omega_k \\ \mathbf{y}_k = L_{\theta_k}\mathbf{x}_k + H_{\theta_k}\eta_k \\ \mathbf{x}_0, \theta_0 \text{ given.} \end{cases} \tag{11}$$

where the noise processes ω_k and η_k are defined in the same way as in (7) and (10), θ_k is called the jump parameter and determines the current system parameter matrices, and $\mathbf{x}_0$ and θ_0 are the given initial conditions of the contour state and the jump parameter. The jump parameter is assumed to evolve stochastically as a Markov chain process with transition matrix T and finite state-space $\mathcal{S} = \{1, \cdots, N\}$.

The MJT is based on an adaptation of the Linear Minimum Mean Square Error (LMMSE) Estimator for Discrete-time Markovian jump linear systems first presented in Costa (1994) (see Fragoso et al. (2005) and Fragoso and Rocha (2006) for continuous-time versions).

In our work, we assume that the current operation mode θ_k is unknown at each time k. In this case, it is well known that the optimal nonlinear filter is obtained from a varying size bank of Kalman filters, which requires exponentially increasing amount of memory and processing with time. Many important *nonlinear sub-optimal* filters were proposed among them the Interactive Multiple Model (IMM) algorithm by Blom and Bar-Shalom (1988) and the one in Doucet et al. (2000). The LMMSE filter is, in contrast, a *optimal-linear* state estimator for the MJLS.

The LMMSE filter works with the following assumptions:

(1) $B_i B_i' > 0$ ($'$ = transpose) for all $i \in \mathcal{S}$.
(2) $\{\omega_k\}$ and $\{\eta_k\}$ are null mean second-order, independent wide sense stationary sequences mutually independent with covariance matrices equal to the identity.
(3) $\mathbf{x}_0\delta_{\{\theta_0=i\}}, i \in \mathcal{S}$ are second order random vectors with the expected value $E(\mathbf{x}_0\delta_{\{\theta_0=i\}}) = \mu_i$ and $E(\mathbf{x}_0\mathbf{x}_0'\delta_{\{\theta_0=i\}}) = V_i, i \in \mathcal{S}$.
(4) $\mathbf{x}_0$ and $\{\theta_k\}$ are independent of $\{\omega_k\}$ and $\{\eta_k\}$.

In fact, the LMMSE filter doesn't estimate the state $\mathbf{x}_k$ directly, instead it estimate the random vector $\mathbf{z}_k$ defined as follows:

$$\mathbf{z}_k^j \triangleq \mathbf{x}_k\delta_{\{\theta_k=j\}} \in \Re^n \tag{12}$$

$$\mathbf{z}_k \triangleq \begin{pmatrix} \mathbf{z}_k^1 \\ \vdots \\ \mathbf{z}_k^n \end{pmatrix} \in \Re^{Nn}. \tag{13}$$

where, in our case, $n = 2N_X$. We also define $\mathbf{q}_k = E(\mathbf{z}_k)$, the projection $\hat{\mathbf{z}}_{k|k-1}$ of $\mathbf{z}_k$ onto the linear subspace spanned by $\mathbf{y}^{k-1} = (\mathbf{y}_{k-1}^* \cdots \mathbf{y}_0^*)$ and

$$\tilde{\mathbf{z}}_{k|k-1} \triangleq \hat{\mathbf{z}}_k - \mathbf{z}_{k|k-1}.$$

The second-moment matrices associated to the above variables are

$$\begin{aligned} Q_k^i &\triangleq E(\mathbf{z_k^i z_k^{i*}}) \in \mathbb{B}(\Re^n), i \in \mathcal{S}, \\ Z_k &\triangleq E(\mathbf{z}_k\mathbf{z}_k^*) = diag[Q_k^i] \in \mathbb{B}(\Re^{Nn}), \\ \hat{Z}_{k|l} &\triangleq E(\hat{\mathbf{z}}_{k|l}\hat{\mathbf{z}}_{k|l}^*) \in \mathbb{B}(\Re^{Nn}), 0 \le l \le k, \\ \tilde{Z}_{k|l} &\triangleq E(\tilde{\mathbf{z}}_{k|l}\tilde{\mathbf{z}}_{k|l}^*) \in \mathbb{B}(\Re^{Nn}), 0 \le l \le k. \end{aligned} \tag{14}$$

where $diag[Q_k^i]$ is the block diagonal matrices with diagonal elements $Q_k^i, \forall i \in \mathcal{S}$. We consider the following augmented matrices

$$\mathsf{A} \triangleq \begin{bmatrix} T_{11}A_1 & \cdots & T_{N1}A_N \\ \vdots & \ddots & \vdots \\ T_{1N}A_1 & \cdots & T_{NN}A_N \end{bmatrix} \in \mathbb{B}(\Re^{Nn}) \tag{15}$$

$$\mathsf{H} \triangleq \left[H_1\pi_1^{1/2} \cdots H_N\pi_N^{1/2} \right] \in \mathbb{B}(\Re^{Nn}, \Re^n) \tag{16}$$

$$\mathsf{L} \triangleq [\, L_1 \cdots L_N \,] \in \mathbb{B}(\Re^{Nn}, \Re^n) \tag{17}$$

$$\begin{aligned} \mathsf{B} &\triangleq diag\left[\left[T_{1j}\pi_1^{1/2}B_1 \cdots T_{Nj}\pi_N^{1/2}B_N \right]\right] \\ &\in \mathbb{B}(\Re^{N^2N\mathbf{x}}, \Re^{Nn}). \end{aligned} \tag{18}$$

where $\pi_i = Prob(\theta_k = i), i \in \mathcal{S}$. With that in hand, it can be shown that, accordingly to Costa (1994), the linear minimal mean squared error state estimative, $\hat{\mathbf{x}}_k$, is given by

$$\hat{\mathbf{x}}_k = \sum_{i=1}^N \hat{\mathbf{z}}_{k|k}^i \tag{19}$$

where $\hat{\mathbf{z}}_{k|k}^i$ satisfies the recursive equation

$$\begin{aligned} \hat{\mathbf{z}}_{k|k} &= \hat{\mathbf{z}}_{k|k-1} + \tilde{Z}_{k|k-1}^i \mathsf{L} * (\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{L}^* + \\ &\quad +\mathsf{HH}^*)^{-1}(\mathbf{y}_k - \mathsf{L}\hat{\mathbf{z}}_{k|k-1}) \end{aligned} \tag{20}$$

$$\hat{\mathbf{z}}_{k|k-1} = \mathsf{A}\hat{\mathbf{z}}_{k-1|k-1}, k \ge 1 \tag{21}$$

$$\hat{\mathbf{z}}_{0|-1} = \mathbf{q}(0)$$

In Costa et al. (2005) was proved that the covariance matrix obtained from the LMMSE can be written in terms of the following recursive Riccati difference equation of dimension Nn that is added with the additional term $\mathfrak{B}(Q_k, k)$ that depends on the second moment matrix of the state variable

$$\begin{aligned} \tilde{Z}_{k+1|k} &= \mathsf{A}\tilde{Z}_{k|k-1}\mathsf{A}^* + \mathfrak{B}(Q_k, k) + \mathsf{BB}^* \\ &\quad -\mathsf{A}\tilde{Z}_{k|k-1}\mathsf{L}^*(\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{L}^* + \mathsf{HH}^*)^{-1} \\ &\quad \times\mathsf{L}\tilde{Z}_{k|k-1}\mathsf{A}^*. \end{aligned} \tag{22}$$

where $\tilde{Z}_{k|k-1}$ is the covariance matrix as defined in (14) $Q_k = (Q_k^1, \cdots, Q_k^N)$ are given by the following recursive equation

$$\begin{aligned} Q_{k+1}^j &= \sum_{i=1}^N T_{ij}(A_i Q_k^i A_i^* + \pi_i B_i B_i^*), \\ Q_0^j &= \mathbb{Q}_0\pi_j, j \in \mathcal{S} \end{aligned} \tag{23}$$

and the functional $\mathfrak{B}(\cdot, k) : \mathbb{H}^n \mapsto \mathbb{B}(\Re^{Nn})$ is defined for $\Upsilon = (\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_N) \in \mathbb{H}^n$ by

$$\mathfrak{B}(\Upsilon, k) \triangleq diag\left[\sum_{i=1}^N T_{ij}A_i\Upsilon_i A_i^*\right] - \mathsf{A}(diag[\Upsilon_i])\mathsf{A}^*. \tag{24}$$

note that the extra term $\mathfrak{B}(\cdot, k)$ would be zero for the case with no jumps reducing the equation (22) to the standard Kalman filter Riccati equation.

4. TRACKER DESIGN AND IMPLEMENTATION

In this section we deal with important aspects of the MJT design and implementation, including: the initialization procedure, the generation of the observation sequence, the method for acquiring the dynamic models (the pairs $(A_i, B_i), i = 1, \cdots, N$) and the contour-state estimation by the LMMSE Markovian jump linear filter.

4.1 Initialization

Each video frame -a digital image- is represented in the computer as a 640×480 matrix of pixels in the RGB color space, that is, each color is decomposed in its red, green and blue components and represented as a 3-dimensional vector.

The MJT uses image measurements based on color features, therefore, prior to actual tracking it is necessary to acquire a color statistical characterization of the object and of the background. Thus, in the first frame of every input video, a number of pixels in the object constituting a representative sample of the object's color is provided. In practice, a small rectangle is placed by hand on the object in the first frame (figure 1*a*), then, the average pixel RGB value of the pixels in that rectangle μ_{obj}, the object's pixel variance σ_{obj} variance, as well as the RGB covariance matrix Σ, are calculated for later use. The whole image average RGB value μ_{bg} is also calculated, which is assumed to be a fair model to the object background.

At the current version the tracker works based on the knowledge of the initial contour $\mathbf{Q}_0$ using it as a *template*. In order to obtain $\mathbf{Q}_0$, again, considering only the first video frame, we use the statistical color information in hand to *segment* the whole image using a nonlinear pixel-wise transformation we designed [4] for this purpose defined by the equation:

$$M_{ij} = \exp\left(-\left(\frac{|\mathbf{i}_{ij} - \mu_{obj}|_{mh}}{\sigma_{obj}}\right)^2\right) \qquad (25)$$

where $\mathbf{i}_{ij}$ is the RGB pixel (i, j) of the input image and the norm subscript mh refers to *Mahalanobis* distance measure which we used as a color dissimilarity measure with good empirical results. The Mahalanobis distance between two random vectors x, y of the same distribution covariance matrix Σ is given by:

$$|x - y|_{mh} = \sqrt{(x-y)^T \Sigma^{-1} (x-y)} \qquad (26)$$

Note that the resulting segmented image M is not in the RGB color space, but is a real matrix with entries in the interval $[0, 1]$. An example of a segmented image obtained in this fashion is shown in figure 1*b*.

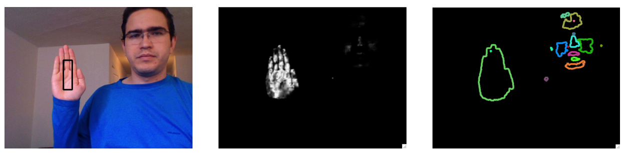
(a) Color sample (b) Segmentation (c) Contours

Fig. 1. Initialization image processing steps.

The final initialization step is the extraction of $\mathbf{Q}_0$ it self, for this we use the standard contour extraction algorithm in the Intel's open computer vision C/C++ library, *OpenCV*, which returns a list of closed contours extracted from the segmented image. In order to select, from the list of extracted contours, the one most likely to be the object's contour we use the fact that it is probably one of the lengthier and is also likely to be located near the sample rectangle earlier mentioned (see figure 1*c*). Recall also that W_0 is obtained from $\mathbf{Q}_0$ by (5).

[4] To the best of the authors' knowledge this nonlinear segmentation transform wasn't proposed previously.

4.2 Observation method

The process of extracting meaningful data from real digital images -the observation process- is a task far from trivial. In fact, solutions to this problem varies largely with the targeted application. One particularity of the application we're working on -tracking- is that for each image a prior estimative of the contour state obtained from the immediately precedent image is available. The measurement method we elected profits from this idiosyncrasy for it traces a number of line normals crossing the mentioned prior contour estimative at evenly separated points. With that accomplished the next step is for each normal the functional

$$\mathcal{J}(t_0) = \int_t |p_i(t) - T_{i,t_0}(t)|^2_{mh} dt \qquad (27)$$

is minimized, for which, $p_i(\cdot)$ is the image RGB pixel value along the ith line normal, t work as an index that takes value equals zero in the inner extreme of the normal and grows linearly up to the total length of the line normal, $T_{i,t_0}(t)$ is a known template built in the following manner:$T_{i,t_0}(t)$ is equal to the average object color for $t \leq t_0$ and $T_{i,t_0}(t)$ is equal to the average background color for $t > t_0$. The value of t_0 that minimizes the functional is probably placed on the current position of the boundary of the object. Line search for feature extraction instead of 2D whole image processing was chosen in order to reduce computational load. The functional in (27) is a generalization for color images of the one presented in Nascimento and Marques (2004).

The next step is to build the actual contour observation $\mathbf{Y}_k \in \Re^{N_X}$ from the above measured feature points, that is, to fit the template to the measurement points. For that purpose we use the method found in chapter six of Blake and Isard (1998). In their work, the curve fitting problem can be stated as a minimization problem, in the following manner: given a prior shape estimate $\mathbf{r}_{k-1}(s)$ (or $\mathbf{X}_{k-1}$ in shape-space) with normals $\mathbf{n}(s)$, and a regularization weight matrix $\bar{S}$ [5], solve:

$$\begin{aligned} &\min_{\mathbf{Y}_k} T \quad \text{where} \\ &T = (\mathbf{Y}_k - \mathbf{X}_{k-1})^T \bar{S} (\mathbf{Y}_k - \mathbf{X}_{k-1}) \\ &\quad + \sum_{i=1}^{N_s} \frac{1}{\sigma_i^2} (\nu_i - \mathbf{h}(s_i)^T [\mathbf{Y}_k - \mathbf{X}_{k-1}])^2 \end{aligned} \qquad (28)$$

where N_s is the number of normals traced in the measurement process and we assume to be also the number of feature points available. We consider therefore the partition $s_i, i \in \mathcal{S}$, s.t. $s_1 = 0, s_{i+1} = s_i + h, s_N = L$ of the curve parameter space $[0, L]$ and let $\mathbf{r}_f(s_i)$ be the feature point

[5] In our case $\bar{S} = W_0^T \mathcal{U} W_0$, where $\mathcal{U} = \frac{1}{L}\int_0^L U^T(s) U(s) ds$.

obtained from the norm passing through precedent curve point $\mathbf{r}_{k-1}(s_i)$ using the measurement method mentioned in the beginning of this section.

That being so, the contour observation is obtained by algorithm 1. Basically, the algorithm iteratively approximate the solution for (28), which is equivalent to iteratively find the contour-configuration $\mathbf{Y}_k$ that fits best to the feature data $\mathbf{r}_f(\cdot)$. The algorithm uses the normal displacement $\nu_i = (\mathbf{r}_f(s_i) - \bar{\mathbf{r}}(s_i))\bar{\mathbf{n}}(s_i)$ instead of the simple displacement $(\mathbf{r}_f(s_i) - \bar{\mathbf{r}}(s_i))$ in order to avoid the burden of variable *curve parameterization* and its effects on the norm $|\mathbf{r}(g(s)) - \mathbf{r}(s)|$, where $g(\cdot)$ is a re-parameterization in $[0, L]$.

Algorithm 1 Curve Fitting

```
$\mathbf{Z}_0 = \mathbf{0}$ and $S_0 = \mathbf{0}$
Iterate, for $i = 1, ..., I$
    $\nu_i = (\mathbf{r}_f(s_i) - \bar{\mathbf{r}}(s_i))\bar{\mathbf{n}}(s_i);$
    $\mathbf{h}(s_i)^T = \bar{\mathbf{n}}(s_i)^T U(s_i) W;$
    $\rho_i = \sqrt{\mathbf{h}(s_i)^T \mathbf{h}(s_i)};$
    if $(|\nu_i| < 2\rho)$ then
        $S_i = S_{i-1} + \frac{1}{\sigma_i^2}\mathbf{h}(s_i)\mathbf{h}(s_i)^T;$
        $\mathbf{Z}_i = \mathbf{Z}_{i-1} + \frac{1}{\sigma_i^2}\mathbf{h}(s_i)\nu_i;$
    Else
        $S_i = S_{i-1}, \ \mathbf{Z}_i = \mathbf{Z}_{i-1};$        (29)
The aggregated observation factor is $\mathbf{Z} = \mathbf{Z}_I$ with associated statistical information $S = S_I$.
Finally, the best fitting curve is given in shape-space by:
    $\mathbf{Y}_k = \mathbf{X}_{k-1} + (\bar{S} + S)^{-1}\mathbf{Z}.$
```

4.3 State Estimation

For the state estimation we make use of the observation state instead of the observation contour configuration, which we chose to be

$$\mathbf{y}_k = \begin{pmatrix} \mathbf{X}_{k-1} \\ \mathbf{Y}_k \end{pmatrix} \tag{30}$$

For our purposes, $L_j = H_j = I_{n\times n}$ for all $j \in \mathcal{S}$ is appropriate. The dynamical model parameters $\{(A_i, B_i)\}_{\{i=1,2,...,N\}}$, however, are *learned* from training data in the process described in the next section. The transition matrix T and the probability vector π are adjusted by hand. The state estimation is done using the equations presented in section 3.

4.4 Parameter Learning

For learning the parameters we provide N short videos with clutter free background, low noise, and deliberately slow motion, so that, the optimality conditions of the Kalman Filter are approximated. The video should be short and representative of a specific kind of motion (rotation, translation, etc.) that is expected to be present in the test videos. We then use the following procedure to acquire the parameters:

(1) For each short video run the single model (Kalman) filter assuming constant velocity default dynamics, that is, using:

$$A_{default} = \begin{pmatrix} \mathbf{0} & \mathbf{1} \\ A^2 & A^1 \end{pmatrix} \tag{31}$$

$$B_{default} = \begin{pmatrix} \mathbf{0} \\ B^0 \end{pmatrix} \tag{32}$$

with $A^1 = -I_{N_X}$, $A^2 = 2I_{N_X}$ and

$$B^0 = \gamma_0 t^{3/2} \left(W_0^T \mathcal{U} W_0\right)^{-\frac{1}{2}}$$

where I_{N_X} is the $N_X \times N_X$ identity matrix, t is the time in seconds and γ_0 is the rate of growth of the radii of spherical search area ($\approx 35 pixels/s$). The estimated *configuration* sequence $\{\mathbf{X}_k\}$ for the interval in which the tracker estimates correctly is recorded, let's say the first J video frames.

(2) The training data obtained in the above step is then employed in the following equations to identify the dynamic parameters A_i and B_i that would *emulate* the $i-th$ training set.

First calculate the auxiliary vectors R_i and matrices R_{ij} and R'_{ij} for $i, j = 0, 1, 2$:

$$R_i = \sum_{k=3}^{J} \mathbf{X}_{k-i}, \quad R_{ij} = \sum_{k=3}^{J} \mathbf{X}_{k-i}\mathbf{X}_{k-j}^T,$$
$$R'_{ij} = R_{ij} - \frac{1}{J-2} R_i R_j^T. \tag{33}$$

Then estimate the subparts of A_i and B_i as in (9):

$$A^2 = \left(R'_{02} - R'_{01}R'^{-1}_{11}R'_{12}\right)\left(R'_{22} - R'_{21}R'^{-1}_{11}R'_{12}\right)^{-1}$$
$$A^1 = \left(R'_{01} - A^2 R'_{21}\right) R'^{-1}_{11}$$
$$B^0 = \left(\frac{1}{J-2}\left(R_{00} - A^2 R_{20} - A^1 R_{10} - \mathbf{D}R_0^T\right)\right)^{\frac{1}{2}}$$

where

$$\mathbf{D} = \frac{1}{J-2}\left(R_0 - A^2 R_2 - A^1 R_1\right).$$

finally, A_i and B_i are obtained in terms of A^2, A^1 and B^0 as defined by (9).

(3) Repeat the process above for the N available exemplar videos .

5. EXPERIMENTS

We initially predicted that using a large number of very *specific* dynamic models would yield a superior performance compared to the usage of a smaller number of more *general* models, therefore, we considered at first different motion models for different directions of translation instead of a single model model for every translation direction. We acted analogously for rotation and scaling ending up with 14 models. However, when the result was compared to the one obtained when the tracker was run with only 3 models (rotation, translation and scaling) the later presented better stability, being able to track the whole test video, i.e. 719 frames, whether the 14-model MJT was able to track accurately for the first 500 frames and suddenly lose the object thereafter.

Another problem with an arbitrary large number of models, aside higher instability, is the computational load. In fact, the 14-models MJT is rather slow ($\approx$ 4 frames per second) compared to the 3-models MJT which executes in real-time (faster than 20 frames per second) on a 2.0GHz notebook with 1GB of memory. In general, the MJT filter working with N modes is roughly N times slower than the single mode Kalman filter and for a small number of dynamic models (≈ 4), our algorithm performed faster and with a smaller memory footprint than our particle filter implementation.

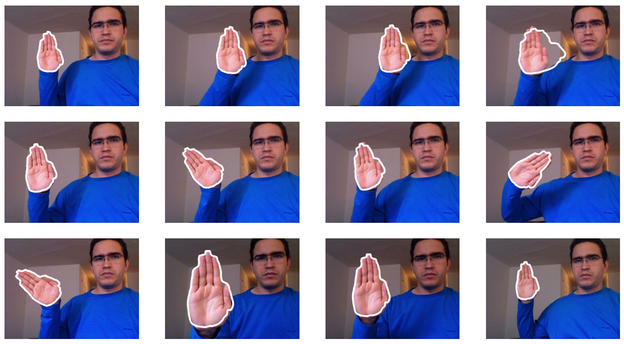

Fig. 2. Sample frames of the output of the MJT with $N = 3$.

Notwithstanding, the MJT (see figure 2), with either 14 or 3 models, has shown to be more stable than the Kalman filter (see figure 3), presenting however a similar tracking accuracy.

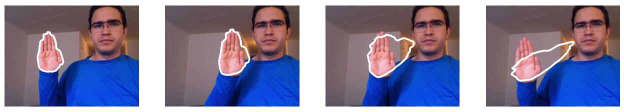

Fig. 3. Sample frames from the output of the Kalman filter. This estimator is accurate when is locked on the object but is very unstable.

The CONDENSATION algorithm (Isard and Blake (1998a)) is a well known adaptation of the *particle filter* for contour tracking. In our experiment the test video presented abrupt changes in the type of motion and , in spite of its robustness, the CONDENSATION algorithm based on a single dynamic model performed poorly (see figure 4) suggesting the necessity of multiple models for this type of motion.

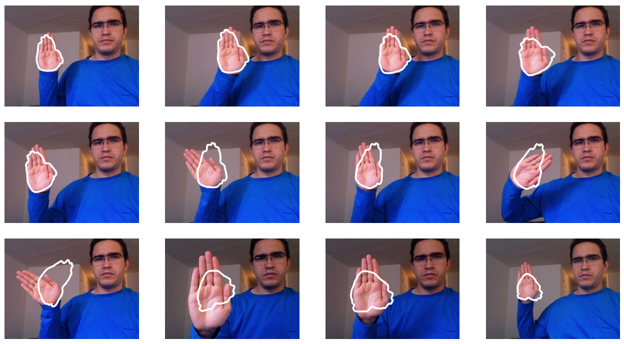

Fig. 4. Sample frames from the video output of the CONDENSATION algorithm.

6. CONCLUSIONS

In this paper we modeled the problem of contour tracking in videos as a MJLS state estimation problem and proposed a solution for it based on the LMMSE state estimator, a non-linear transformation for image segmentation, and other concepts. Our method, named MJT for short, presented a superior stability compared to the Kalman filter and clearly a greater precision compared to the standard particle filter in our preliminary experiments.

REFERENCES

V. Athitsos and S. Sclaroff. An appearance-based framework for 3D hand shape classification andcamera viewpoint estimation. *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 40–45, 2002.

A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion.* Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1998.

H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with Markovianswitching coefficients. *Automatic Control, IEEE Transactions on*, 33(8):780–783, 1988.

H.A.P. Blom and EA Bloem. Particle filtering for stochastic hybrid systems. *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, 3, 2004.

O.L.V. Costa. Linear minimum mean square error estimation for discrete-timeMarkovian jump linear systems. *Automatic Control, IEEE Transactions on*, 39(8):1685–1689, 1994.

O.L.V. Costa, M.D. Fragoso, and R.P. Marques. *Discrete-time Markov Jump Linear Systems.* Springer, 2005.

A. Doucet, A. Logothetis, and V. Krishnamurthy. Stochastic sampling algorithms for state estimation of jump Markovlinear systems. *Automatic Control, IEEE Transactions on*, 45(2):188–202, 2000.

M.D. Fragoso and N.C.S. Rocha. Stationary filter for continuous-time markovian jump linear systems. *SIAM Journal on Control and Optimization*, 44(6):801–815, 2006.

M.D. Fragoso, O.L.V. Costa, J. Baczynski, and N.C.S. Rocha. Optimal linear mean square filter for continuous-time jump linear systems. *IEEE Transactions on Automatic Control*, 50(9):1364–1369, September 2005.

M. Isard and A. Blake. CONDENSATION—Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998a.

M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. *Computer Vision, 1998. Sixth International Conference on*, pages 107–112, 1998b.

J.C. Nascimento and J.S. Marques. Robust shape tracking in the presence of cluttered background. *Multimedia, IEEE Transactions on*, 6(6):852–861, 2004.

C. Tomasi, S. Petrov, and A. Sastry. 3d tracking = classification + interpolation. In *Ninth IEEE International Conference on Computer Vision (ICCV'03)*, 2003.