Tracking Nonparameterized Object Contours in Video

Hieu Tat Nguyen, Marcel Worring, Rein van den Boomgaard, and Arnold W. M. Smeulders

***Abstract*—We propose a new method for contour tracking in video. The inverted distance transform of the edge map is used as an edge indicator function for contour detection. Using the concept of topographical distance, the watershed segmentation can be formulated as a minimization. This new viewpoint gives a way to combine the results of the watershed algorithm on different surfaces. In particular, our algorithm determines the contour as a combination of the current edge map and the contour, predicted from the tracking result in the previous frame. We also show that the problem of background clutter can be relaxed by taking the object motion into account. The compensation with object motion allows to detect and remove spurious edges in background. The experimental results confirm the expected advantages of the proposed method over the existing approaches.**

***Index Terms*—Contour tracking, motion analysis, watershed algorithm.**

I. Introduction

THIS PAPER addresses the problem of tracking the contour of a moving object in video. In essence, every iteration of a contour tracking procedure consists of updating the contour based on measurements in the current frame, taking into account the results from previous iterations.

According to the method used for contour detection, we classify tracking methods in literature into two categories depending on whether the methods track a parameterized contour or a nonparameterized contour.

Most tracking methods belong to the first class where the contour is approximated by a parametric model. For example, in the methods of Blake and Isard [1], [2], the contour is approximated by B-splines. The methods first fit a B-spline curve to intensity edges and then use a Kalman filter or a Monte-Carlo filter to track the B-spline coefficients. In [3], Fourier coefficients are used as parameters of the contour. In several other tracking algorithms [4], [5], the contour detection is performed by minimizing some energy function, using extensions from the snake model of Kass *et al.* [6]. In the Kalman snakes, the Euler–Lagrange equation is the basis for the construction of the predicted next instance of the contour. The implementation of the model requires the contour be approximated by a polygon with a fixed number of vertices.

Manuscript received December 5, 2000; revised May 6, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Scott T. Acton.

The authors are with the Intelligent Sensory Information Systems Group, Faculty of Science, University of Amsterdam, NL-1098 SJ, Amsterdam, The Netherlands (e-mail: tat@science.uva.nl; worring@science.uva.nl; rein@science.uva.nl; smeulders@science.uva.nl).

Publisher Item Identifier 10.1109/TIP.2002.802522.

In all of these methods, the state of the object is represented by a fixed number of parameters characterizing the contour and its motion. One great advantage of this approach is the possibility of modeling the tracking as an estimation problem for a hidden Markov model. Powerful estimation tools associated with this model such as the Kalman filter [7] or Monte-Carlo filters [8] can be used to track the parameter values over the sequence. Snake parameterization can also be well adjusted for global as well as local behavior of the snake once some knowledge about the object shape is available. While some papers indicate the difficulty of dealing with topological changes like swallow tails [9] or cusps [3], this, in fact, can be handled by explicitly taking care of any collision of the contour with itself or with other contours, and reinitializing the contour after the collision [10]. G-snakes in [11] handle local contour deformations based on learning from a set of training examples. The absence of domain knowledge about the object shape, however, may cause inaccuracies in the tracking, especially when the motion is nonrigid.

In the methods of the second class, the contour is represented as a border of a region [12], [13]. The strength of this approach is that it allows for contour representation of an arbitrary shape. Furthermore, the contour topology can be easily controlled. The geodesic contour model [14] belongs to this class. It is used by Paragios *et al.* in [12] to track objects moving against a static background. The method employs a modified level-set algorithm to propagate the contour toward the edges. The method, however, will fail when the camera is moving. In [13], the tracking is performed by means of motion segmentation. The watershed algorithm is employed to segment three-dimensional (3-D) optic flow. As optic flow estimation around object boundaries is usually inaccurate, the contour obtained with the watershed algorithm is also inaccurate. In addition, since the method is sensitive to segmentation errors in individual frames, the tracking result is unstable.

The goal of this paper is to develop a new method that tracks nonparameterized contours. At the same time, the new method should have the following desirable properties:

1) the method should have the ability to track contours under the condition of a moving camera and the presence of multiple moving objects;
2) the method should be able to track fast moving objects and should deal with abrupt motion changes;
3) the method should be able to track nonrigid objects;
4) the method should be robust to missing edges;
5) the method should be robust to image clutter.

To the best of our knowledge, none of the methods in literature fulfills all these reasonable requirements.

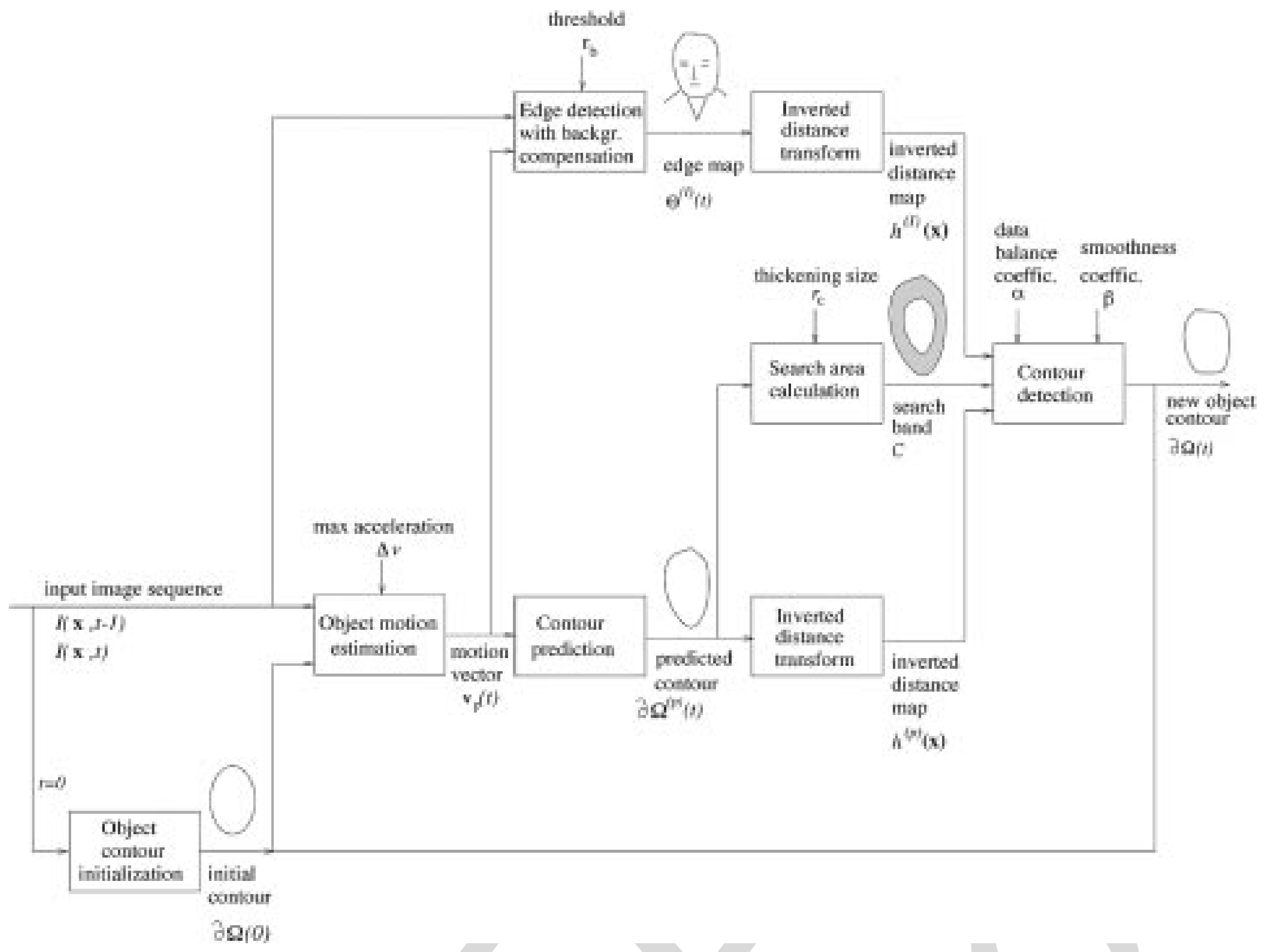

Fig. 1. Illustration for the tracking scheme.

This paper is structured as follows. In Section II we describe the general tracking scheme. Section III describes the steps required before the detection of the object contour. An algorithm for object contour detection is presented in Section IV. Tracking results are shown in Section V. Section VI discusses the parameter setting and the properties of the method with a verification of the above requirements.

II. Overview of the Tracking Scheme

This section gives the description for the general tracking scheme, depicted in Fig. 1.

Let $\Omega(t)$ be the image region occupied by the object at the current time t. We want to determine the object contour $\partial\Omega(t)$, given the contour $\partial\Omega(t-1)$ obtained from the tracking in the previous frame.

First, the position of the object is predicted for the current frame. To that end, we estimate the motion of the object and warp $\partial\Omega(t-1)$ into the current frame. Let $\partial\Omega^{(p)}$ be the predicted contour. When the prediction is accurate, the real object contour $\partial\Omega(t)$ differs only marginally in shape and location from the predicted contour $\partial\Omega^{(p)}$. Hence, $\partial\Omega(t)$ is confined to a band $\mathcal{C}$ around $\partial\Omega^{(p)}$. We construct $\mathcal{C}$ by thickening $\partial\Omega^{(p)}$. The edge map is then computed for the current frame. Using object motion to the full, irrelevant edges in background are removed. Within $\mathcal{C}$, $\partial\Omega(t)$ is searched for while taking into account closeness to the current edge map $\Theta^{(I)}$ as well as the predicted contour $\partial\Omega^{(p)}$.

Finally, the object contour $\partial\Omega(t)$ is obtained by minimizing an energy function. The process is then repeated for the next frame.

Every step in this scheme is elaborated upon in the two next sections.

III. Extraction of Edge Indicator Functions

This section describes the steps taken to obtain the edge indicator functions, required for contour detection.

A. Object Contour Initialization

The object contour is bootstrapped by segmentation of the first frame. We use the method in [15] which performs a color segmentation, followed by merging regions undergoing similar motions. The merging yields accurate results when the object motion is well described by polynomial models such as affine or quadratic models. In general, this is true, when the moving object is rigid and distant enough from the camera. When this is not the case, the method may not merge all regions on the object but it reduces the number of regions substantially. The object extraction then requires some user interaction.

B. Object Motion Estimation and Contour Prediction

To obtain the predicted contour $\partial\Omega^{(p)}$, we first estimate the object motion between two frames $I(\mathbf{x},\, t-1)$ and $I(\mathbf{x},\, t)$ using a parametric motion model, and then warp the previous contour $\partial\Omega(t-1)$ onto the current frame $I(\mathbf{x},\, t)$. In this paper, a translation model is used. The dominant translation vector $\mathbf{v}_p(t)$ is estimated by minimizing the motion-compensated prediction error, formulated as a sum-of-squared differences

$$\mathbf{v}_p(t) = \arg\min_{\mathbf{v}\in\mathcal{V}} \sum_{\mathbf{x}\in\Omega(t-1)} [I(\mathbf{x},\, t-1) - I(\mathbf{x}+\mathbf{v},\, t)]^2 \quad (1)$$

where $\mathcal{V}$ is the velocity space. Thus, the object motion is estimated by matching intensities in two consecutive frames, and without computing an optic flow field beforehand. The contour $\partial\Omega^{(p)}$ is then obtained by translating $\partial\Omega(t-1)$ over $\mathbf{v}_p(t)$.

One may be concerned about the limitation of the simple translation model for describing rotation, zooming and nonrigid motion. Nevertheless, this limitation does not cause problems for the tracking algorithm. Since the magnitude of nontranslational motion between two consecutive frames is usually small, the local contour deformations, which cannot be explained by pure translation, are later captured in the step of contour detection.

In practice, vector $\mathbf{v}_p(t)$ is found by an exhaustive search in the discrete rectangular $\mathcal{V} = [-v_{x\,\max},\, v_{x\,\max}]\times[-v_{y\,\max},\, v_{y\,\max}]$, where $v_{x\,\max}$ and $v_{y\,\max}$ are the maximum speeds of the object in x and y directions in the image plane respectively. $v_{x\,\max}$ and $v_{y\,\max}$ are set according to the object motion vector, estimated in the previous frame with margin Δv

$$\begin{aligned} v_{x\,\max} &= v_{xp}(t-1) + \Delta v \\ v_{y\,\max} &= v_{yp}(t-1) + \Delta v \end{aligned} \quad (2)$$

where $v_{xp}(t-1)$ and $v_{yp}(t-1)$ denote the two components of $\mathbf{v}_p(t-1)$, and Δv is the maximal margin between the current motion vector and the previous one due to acceleration or abrupt changes in direction of motion. Note that the chance of an abrupt change in motion direction is usually higher than the chance of an abrupt change in speed magnitude. Therefore, we center $\mathcal{V}$ at $\mathbf{0}$ rather than $\mathbf{v}_p(t-1)$ in order to capture abrupt changes in motion direction.

For the sake of efficiency and accuracy, searching for the velocity is done in a multiscale manner. Initially, the search is carried out at a coarse resolution using exhaustive search. As the exhaustive search finds the global minimum of (1), it allows for robust object motion estimation even in case of fast translations. At higher resolutions, $\mathbf{v}_p(t)$ is searched for only around the result found at the lower resolution.

The object motion estimated is used not only for contour prediction but also for the removal of background edges in Section III-D. Note also that the proposed method does not need to estimate the camera motion or the motion of other moving objects in the scene.

C. Search Area Construction

The search band $\mathcal{C}$ is constructed by thickening $\partial\Omega^{(p)}$. We use the homotopic thickening [16] so that $\mathcal{C}$ is homotopic to $\partial\Omega^{(p)}$. This implies that the band $\mathcal{C}$ separates the rest of the image into two connected regions: an interior $\mathcal{M}_{int}$ and an exterior $\mathcal{M}_{ext}$. If the size of the thickening r_c is large enough, we assign $\mathcal{M}_{int}$ to the object and $\mathcal{M}_{ext}$ to the background. Hence, only pixels inside the band need to be addressed. Recommendations for setting r_c are given in Section V-A.

D. Edge Detection and Background Edge Removal

This subsection describes edge detection and the removal of irrelevant edges in background.

We employ the Canny edge detector to detect the intensity edges. We keep the significant edges, eliminating those whose gradient magnitude is lower than a given threshold T_c. The setting of T_c depends on the contrast at the object contour. In general, we prefer low values of T_c to guarantee the presence of the entire object contour in the edge map, although low values of T_c produce more spurious edges. We use $T_c = 1.0$. Let $\Theta^{(I)}(t)$ be the edge map detected in frame $I(\mathbf{x},\, t)$.

Contour detection may be affected by irrelevant edges, especially those in the background which have been occluded previously. When object motion is known approximately, we can tell the background edges from the object edges as they have different motion vectors. For this purpose, we project the whole edge map of the previous frame $\Theta^{(I)}(t-1)$, including edges of the background and other moving objects, into the current frame $I(\mathbf{x},\, t)$ according to the object motion vector $\mathbf{v}_p(t)$. Object edges will be projected onto themselves while edges of the background and other moving objects will not. Thus, the latter irrelevant edges can be found by looking at the discrepancies between the projected edge map $\Theta_w(t)$ and the actual edge map $\Theta^{(I)}(t)$. We first compute the distance transform of $\Theta_w(t)$ and look for pixels in $\Theta^{(I)}(t)$ for which the distance exceeds a threshold r_b

$$\Theta_{background}(t) = \left\{ \mathbf{x}\in\Theta^{(I)}(t) \,\middle|\, d(\mathbf{x},\, \Theta_w(t)) > r_b \right\} \quad (3)$$

where $\Theta_{background}(t)$ denotes the set of background edges in $\Theta^{(I)}(t)$. Results for real data are shown in Fig. 3.

Note that some background edges persist in the following cases:

1) Θ_w and $\Theta^{(I)}(t)$ intersect accidentally;
2) an edge segment has the same direction as $\mathbf{v}_p(t)$, and a length exceeding the length of $\mathbf{v}_p(t)$.

Such edge segments, nevertheless, are usually small and isolated, and hence, have little influence on the result.

For simplicity of notations, in the rest of the paper we also use $\Theta^{(I)}(t)$ to denote the edge map resulting after background edges are removed.

E. Inverted Distance Transform of the Edge Map and the Predicted Contour

This subsection derives the edge indicator function, required for contour detection.

The common choice for edge indicator functions is the intensity gradient. In practice, however, irrelevant edges in the background or inside the object may have a higher gradient magnitude than the object contour. For still images, this is not a problem if the initial contour is placed close to the true contour.

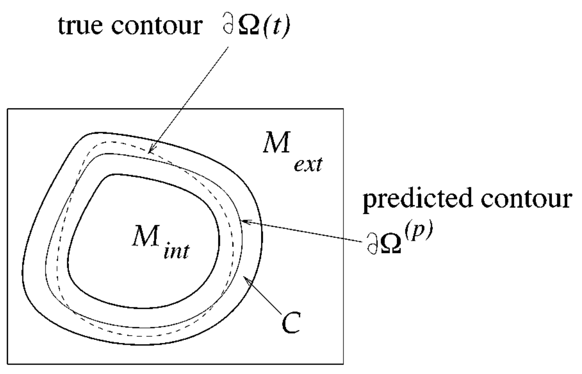

Fig. 2. Illustration of the calculation of the search area $\mathcal{C}$.

In image sequences, however, the gradient at the contour varies with time. A strictly gradient-based approach, therefore, leads to unstable tracking results as it may attract the contour toward irrelevant edges with high gradient values.

To overcome the problem, we need an edge indicator function invariant to object motion. We choose the inverted distance transform of the edge map [17], [18]

$$h^{(I)}(\mathbf{x}) = M - d\left(\mathbf{x},\, \Theta^{(I)}\right) \tag{4}$$

where $d(\mathbf{x},\, \Theta^{(I)})$ is the Euclidean distance from $\mathbf{x}$ to $\Theta^{(I)}$, and M is an arbitrary constant such that $h^{(I)}(\mathbf{x}) > 0$ everywhere. An algorithm for computing $d(\mathbf{x},\, \Theta^{(I)})$ is given in [19]. Three-dimensional views of this function for some simple edge maps are shown in Fig. 4. Since the inverted distance transform has a constant gradient almost everywhere, it can guide a contour moving to edges over a long distance.

In order to increase tracking stability, $\partial\Omega(t)$ should be close to the predicted contour $\partial\Omega^{(p)}$. Therefore, we use the inverted distance transform of the predicted contour as well

$$h^{(p)}(\mathbf{x}) = M - d\left(\mathbf{x},\, \partial\Omega^{(p)}\right). \tag{5}$$

Both $h^{(I)}(\mathbf{x})$ and $h^{(p)}(\mathbf{x})$ are exploited in the next section to find the object contour $\partial\Omega(t)$.

IV. Contour Detection

We now move to the most important step in the tracking process: the detection of the object contour $\partial\Omega(t)$.

For contour detection, we employ the watershed algorithm from mathematical morphology [17], [20], using the two regions $\mathcal{M}_{int}$ and $\mathcal{M}_{ext}$ as markers. The advantage of using the watershed algorithm on a set of two markers is that the resulting segmentation always contains only two regions and the resulting contour is always closed and simple. Furthermore, the contour is positioned at the most significant edges within the search band $\mathcal{C}$. It can be proven that the watershed algorithm yields the precise contour under certain regularity conditions. The problem yet to be solved is that the original watershed algorithm takes as input only one edge indicator function. We want to use two edge indicator functions: the inverted distance transform of the current edge map $h^{(I)}(\mathbf{x})$ and the inverted distance transform of the predicted contour $h^{(p)}(\mathbf{x})$ in combination to increase the robustness of the contour detection as a compromise between the current edge map and the predicted contour. Hence, a new algorithm with two edge indicator functions needs to be developed.

Before describing the actual algorithm, it is helpful to consider how the watershed algorithm is applied in case only one edge indicator function is used.

A. Definition of the Watershed

This subsection gives the definition of the watershed algorithm applied for one relief function $h(\mathbf{x})$ using two markers $\mathcal{M}_{int}$ and $\mathcal{M}_{ext}$.

According to [20], in case of the watershed with imposed markers, the original relief is reconstructed. This reconstruction makes the markers sole regional minima of the reconstructed relief. Let $f(\mathbf{x})$ be the reconstructed relief. The function $f(\mathbf{x})$ can be defined via the recursive conditional erosion as proposed in [20]. $f(\mathbf{x})$ can also be defined in a more elegant way as follows [21]:

$$f(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in \mathcal{M}_{int} \cup \mathcal{M}_{ext} \\ \min\limits_{\gamma \in [\mathbf{x} \rightsquigarrow \mathcal{M}_{int} \cup \mathcal{M}_{ext}]} \max\limits_{z \in \gamma} h(z), & \text{if } \mathbf{x} \in \mathcal{C} \end{cases} \tag{6}$$

where $[\mathbf{x} \rightsquigarrow \mathcal{M}_{int} \cup \mathcal{M}_{ext}]$ denotes the set of all possible paths γ from $\mathbf{x}$ to the two markers. Thus, $f(\mathbf{x})$ can be interpreted as the maximal level the water from the markers has to reach before flooding $\mathbf{x}$. For algorithms computing $f(\mathbf{x})$, we refer to [20] and [22].

The object contour is now detected as the watershed line $WS(f)$ of the relief f, viewed as a mountain landscape. $WS(f)$ can be defined formally using the concept of topographical distance [23], [24]. The topographical distance with respect to a smooth surface f is defined as the distance weighted with the gradient norm $|\nabla f|$. However, as the function f is nonsmooth in our case, the following definition due to Meyer [23] is required.

Definition 1: Suppose f is a function defined on $\mathbb{R}^2$. Given a path γ: $[0,\, 1] \mapsto \mathbb{R}^2$ connecting two points $\mathbf{x}$ and $\mathbf{y}$. Consider all possible partitions of γ: $\zeta = (\gamma_1,\, \gamma_2,\, \ldots,\, \gamma_n)$ where $\gamma_i = \gamma(t_i), t_1 < t_2 < \cdots < t_n, t_1 = 0,\, t_n = 1$. Let ε_i be the erosion of f by a disk of radius $|\gamma_{i-1} - \gamma_i|$. The topographical variation of f on γ is defined as

$$TV(\gamma) = \sup_{\zeta} \sum_{i=2}^{n} [f(\gamma_i) - \varepsilon_i f(\gamma_i)] \tag{7}$$

where the supremum is taken over all possible partitions of γ.

Definition 2: The topographical distance between two points $\mathbf{x}$ and $\mathbf{y}$ is the minimal topographical variation over all paths from $\mathbf{x}$ to $\mathbf{y}$

$$L(\mathbf{x},\, \mathbf{y}) = \inf_{\gamma \in [\mathbf{x} \rightsquigarrow \mathbf{y}]} TV(\gamma). \tag{8}$$

Definition 3: The topographical distance from a point $\mathbf{x}$ to a set $\mathcal{M}$ is

$$L(\mathbf{x},\, \mathcal{M}) = \inf_{\mathbf{y} \in \mathcal{M}} L(\mathbf{x},\, \mathbf{y}). \tag{9}$$

An algorithm for the computation of the topographical distance is given in [23].

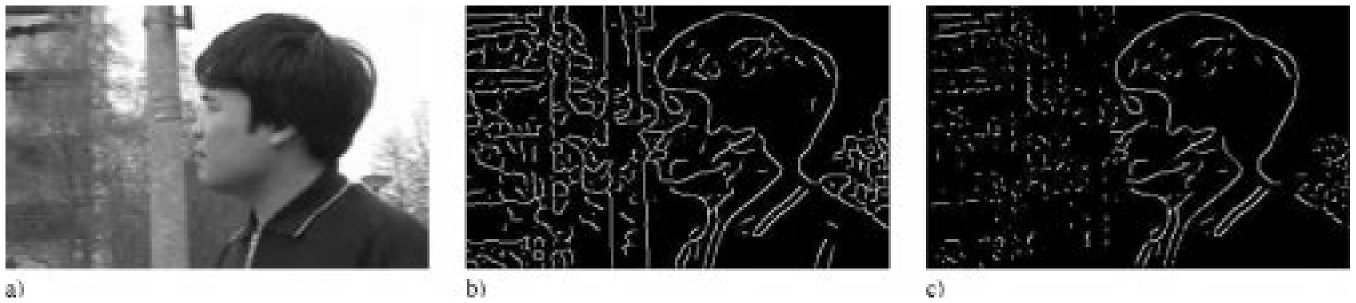

Fig. 3. (a) One frame. The object being tracked is the head of a man, (b) result of the Canny detector, and (c) removal of background edges according to (3).

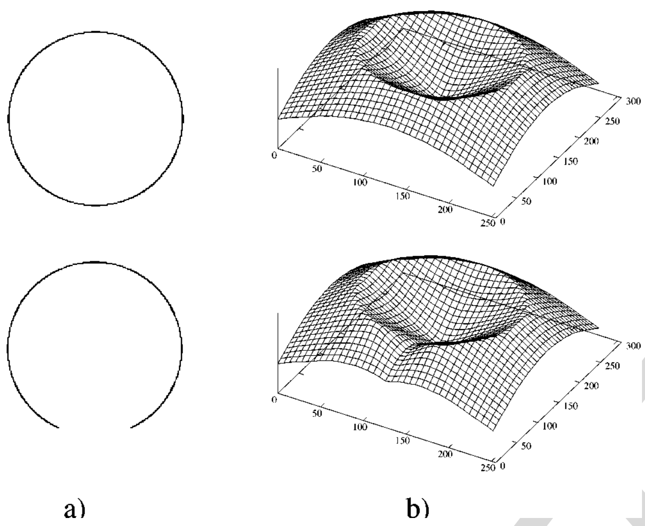

Fig. 4. (a) Some simple edge maps and (b) a 3-D view of the inverted distance function $h(\mathbf{x})$.

Now, let $L_{int}(\mathbf{x})$ and $L_{ext}(\mathbf{x})$ be the topographical distances with respect to the surface f from $\mathbf{x}$ to $\mathcal{M}_{int}$ and $\mathcal{M}_{ext}$, respectively. $WS(f)$ can then be defined as

$$WS(f) = \{\mathbf{x} | L_{int}(\mathbf{x}) = L_{ext}(\mathbf{x})\}. \tag{10}$$

The following sets

$$\begin{aligned} CB_{int}(f) &= \{\mathbf{x} | L_{int}(\mathbf{x}) < L_{ext}(\mathbf{x})\} \\ CB_{ext}(f) &= \{\mathbf{x} | L_{int}(\mathbf{x}) > L_{ext}(\mathbf{x})\} \end{aligned} \tag{11}$$

are the catchment basins of $\mathcal{M}_{int}$ and $\mathcal{M}_{ext}$, respectively.

The definitions of the topographical distance and watershed are used in the next subsection for the combination of the watershed lines of two different surfaces.

B. Combine Measured Edges With the Predicted Contour

This subsection presents an algorithm for finding the object contour $\partial\Omega(t)$, taking into account closeness to both the current edge maps $\Theta^{(I)}(t)$ and the predicted contour $\partial\Omega^{(p)}$.

A summary of the algorithm is given in Fig. 5.

First, the reconstruction is performed for each of the two edge indicator functions $h^{(I)}(\mathbf{x})$ and $h^{(p)}(\mathbf{x})$ according to (6). Let $f^{(I)}$ and $f^{(p)}$ be the two reconstructed relief functions. For each $f^{(\cdot)}$, we compute two topographical distance transforms to $\mathcal{M}_{int}$ and $\mathcal{M}_{ext}$ denoted $L^{(\cdot)}_{int}(\mathbf{x})$ and $L^{(\cdot)}_{ext}(\mathbf{x})$, respectively.

In [21], we prove that the watershed segmentation can be represented as a minimization. In our case, the watershed segmentation with respect to the relief function $f^{(\cdot)}$ can be obtained by minimizing the energy function

$$E^{(\cdot)}(\Omega_{int}) = \iint\limits_{\Omega_{int}} L^{(\cdot)}_{int}(\mathbf{x})\, d\mathbf{x} + \iint\limits_{\Omega_{ext}} L^{(\cdot)}_{ext}(\mathbf{x})\, d\mathbf{x} \tag{12}$$

where Ω_{int} denotes the interior of the object contour and Ω_{ext} its complement. Note that $\iint_{\Omega_{ext}} L^{(\cdot)}_{ext}(\mathbf{x})\, d\mathbf{x} = S - \iint_{\Omega_{int}} L^{(\cdot)}_{ext}(\mathbf{x})\, d\mathbf{x}$, where $S = \iint_{\Omega_{int} \cup \Omega_{ext}} L^{(\cdot)}_{ext}(\mathbf{x})\, d\mathbf{x}$ is a constant, independent of Ω_{int}. Ignoring S, this energy can be rewritten as

$$E^{(t)} = \iint\limits_{\Omega_{int}} \left\{ L^{(\cdot)}_{int}(\mathbf{x}) - L^{(\cdot)}_{ext}(\mathbf{x}) \right\} d\mathbf{x}. \tag{13}$$

The minimization of $E^{(\cdot)}$ is carried out over all possible configurations of Ω_{int}

$$CB_{int}\left(f^{(\cdot)}\right) = \arg\min_{\Omega_{int}} E^{(\cdot)}. \tag{14}$$

Obviously, to find the region Ω_{int} that minimizes this integral we need to collect points where $L^{(\cdot)}_{int}(\mathbf{x}) - L^{(\cdot)}_{ext}(\mathbf{x}) < 0$. The result region, therefore, includes the interior catchment basin $CB_{int}(f^{(\cdot)})$, but does not intersect the exterior catchment basin $CB_{ext}(f^{(\cdot)})$. Its border, where $L^{(\cdot)}_{int}(\mathbf{x}) - L^{(\cdot)}_{ext}(\mathbf{x}) = 0$, coincides with the watershed line $WS(f^{(\cdot)})$.

To find a contour which is a tradeoff between the watershed lines on the surfaces $f^{(I)}$ and $f^{(p)}$, we minimize the following weighted sum of the two energy functions:

$$\begin{aligned} E &= \alpha E^{(I)} + (1-\alpha) E^{(p)} \\ &= \iint\limits_{\Omega_{int}} \left\{ \alpha \left[L^{(I)}_{int}(\mathbf{x}) - L^{(I)}_{ext}(\mathbf{x}) \right] \right. \\ &\qquad \left. + (1-\alpha) \left[L^{(p)}_{int}(\mathbf{x}) - L^{(p)}_{ext}(\mathbf{x}) \right] \right\} d\mathbf{x} \end{aligned} \tag{15}$$

where $\alpha \in [0,\, 1]$ is a data balance coefficient. Thus,

$$\Omega(t) = \arg\min_{\Omega_{int}} E. \tag{16}$$

The minimization is carried out by selecting from the search band $\mathcal{C}$, defined in Section III-C, the points, where the integrand

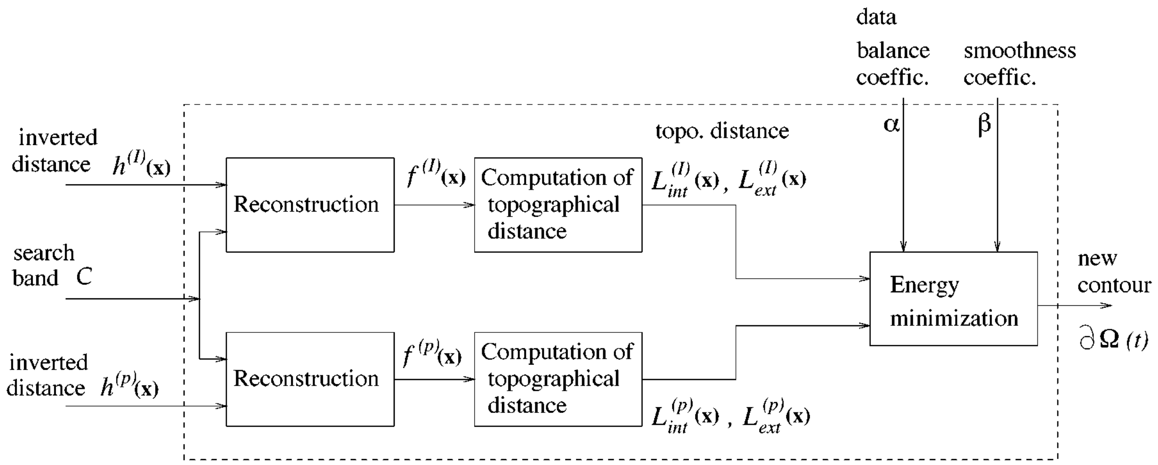

Fig. 5. Illustration for the detection of $\partial\Omega(t)$, a zoomed-in view of the contour detection block in Fig. 1.

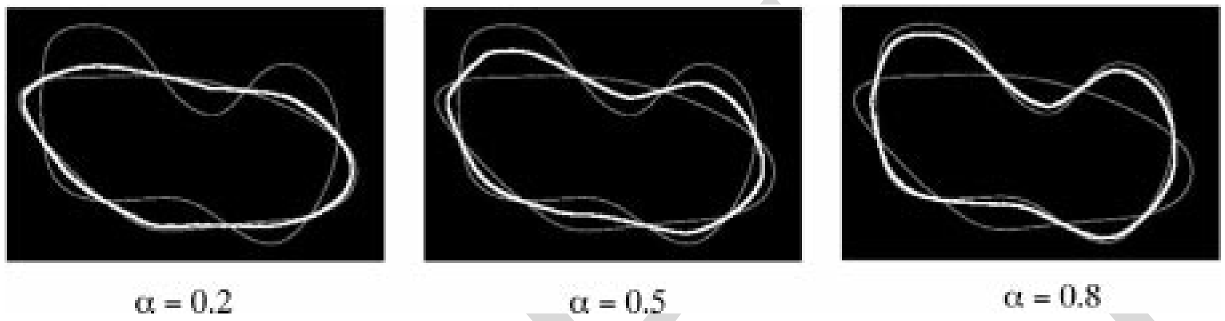

Fig. 6. Illustration of the combination of watershed lines. This experiment used two edge maps which were created from the two thin contours with low brightness. The thick and highlighted contour is the result of the minimization of the energy function (15).

is negative, and adding them to the interior marker $\mathcal{M}_{int}$. The contour of the result region satisfies

$$\alpha\left[L^{(I)}_{int}(\mathbf{x})-L^{(I)}_{ext}(\mathbf{x})\right]+(1-\alpha)\left[L^{(p)}_{int}(\mathbf{x})-L^{(p)}_{ext}(\mathbf{x})\right]=0. \tag{17}$$

The resulting contour belongs to the following set: $[[CB_{int}(f^{(I)})\cap CB_{int}(f^{(p)})]\cup[CB_{ext}(f^{(I)})\cap CB_{ext}(f^{(p)})]]^c$. As a consequence, it lies between the two watershed lines $WS(f^{(I)})$ and $WS(f^{(p)})$ (see Fig. 6). The coefficient α is used to control the influence of the predicted contour to the result contour. As observed from (17), when $\alpha=1$ the contour coincides with the watershed of the surface $f^{(I)}$. When $\alpha<0.5$, the predicted contour has more influence. In this case, the tracker tends to stick to the predicted contour, leading to smooth changes of the tracking results. This, however, prevents tracking nonrigid motion of the contour.

The incorporation of the predicted contour into result also resolves the problem of missing edges. When the object moves into a region where there is no edge evidence between object and background, the tracker keeps the contour as predicted.

C. Imposing Smoothness

The watershed algorithm itself does not take the smoothness of the contour into account. As a consequence, often the watershed line looks ragged. When smoothness of the contour is required, the contour length is added to the energy function in (15) in order to make the contour "harder" against deformation

$$\min_{\Omega_{int}}\left\{\alpha E^{(I)}+(1-\alpha)E^{(p)}+\beta\beta(t)\int_{\partial\Omega_{int}}ds\right\} \tag{18}$$

where β is a constant and $\beta(t)$ is a coefficient which does not depend on Ω_{int}. The coefficient $\beta(t)$ is meant to make the tuning of β invariant to scaling. Observe that when the image is enlarged n times, the values of $E^{(I)}$ and $E^{(p)}$ increase n^3 times while the contour length increases n times only. Therefore, we use

$$\beta(t)=s^2(t-1) \tag{19}$$

where $s(t-1)$ is the maximal size of the bounding box of the object region $\Omega(t-1)$ in the previous frame.

Starting from the segmentation result of the previous subsection, the smoothing stage performs an exchange of pixels at the border between Ω_{int} and Ω_{ext} such that the energy (18) is minimized. For the detailed implementation we refer to [21].

In conclusion, the object contour $\partial\Omega(t)$ is obtained by minimizing an area functional, computed from the topographical distances to the markers with respect to the reconstructed edge indicator surfaces $f^{(I)}(\mathbf{x})$ and $f^{(p)}(\mathbf{x})$. The minimization of this energy function yields a tradeoff between the watershed lines on the two surfaces $f^{(I)}(\mathbf{x})$ and $f^{(p)}(\mathbf{x})$. Smoothness of the contour, when desired, can be imposed by adding the contour length to the energy.

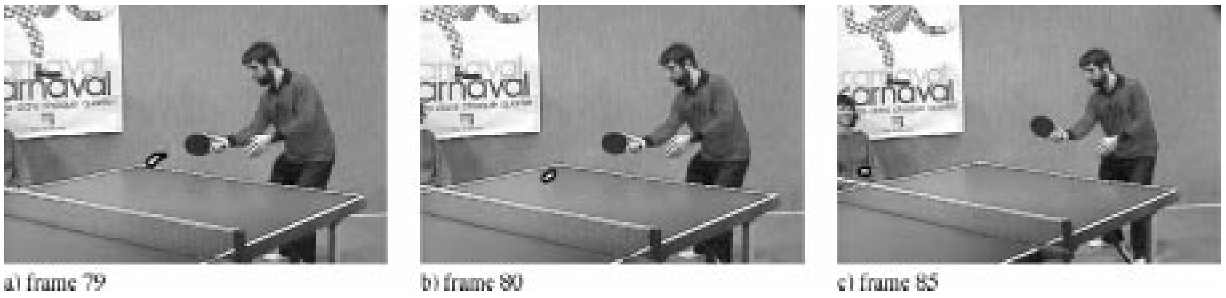

Fig. 7. Results of tracking the tennis ball in the table tennis sequence (marked by a black outline).

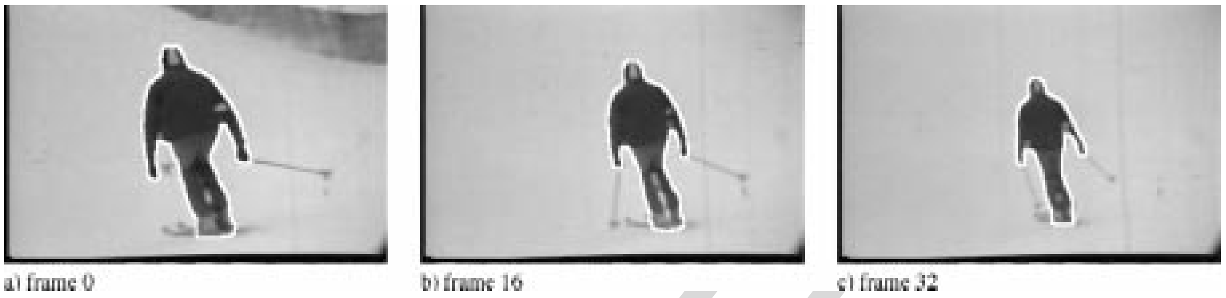

Fig. 8. Tracking results.

V. EXPERIMENTS

A. Parameter Setting

Let us first discuss the dependence of the algorithm on the parameters.

1) *The thickening size r_c*

In case of a clear object in a clear background, we should just set r_c larger than the expected maximal displacement between the true contour and the predicted contour. The displacement is caused by the nonrigidness of object and its limbs. In case of a cluttered object or background, larger values of r_c increase the number of spurious edges in the search band, which increases the risk of following wrong edges. We use $r_c = 4h$, where h is the pixel size.

2) *The data balance coefficient α*

The effect of tuning α has been discussed in Section IV-B. In case of a rigid object with a simple motion, one may trust the predicted contour $\partial\Omega^{(p)}$ more than the current edge map $\Theta^{(I)}(t)$, and therefore use high values of α to emphasize the influence of $\partial\Omega^{(p)}$. In case of nonrigid objects or complex motion, since the prediction may not always be accurate, low values of α are preferred to stress $\Theta^{(I)}(t)$. We assume equal influence from the past and the data by setting: $\alpha = 0.5$.

3) *The threshold r_b for background edge compensation*

For the usual background we set the value of r_b equal to r_c because they both are due to the accuracy of the contour prediction. In order not to loose too much elements of the real contour, it is better to use a conservative value of r_b. We use $r_b = 1\,h$.

4) *Maximal margin between the current speed and the previous one Δv*

Δv should take large values in sequences where the magnitude of object speed changes abruptly, and small values when the object motion is smooth. Note that a large value of Δv is always safe but such a value leads to high computation costs. We use $\Delta v = 3\,h$.

5) *Smoothness coefficient β*

Contour smoothing illustrates the ability to incorporate *a priori* knowledge about the object shape into the resulting contour. The value of β should be given a value compatible with derivative of the first two terms in eq. (18) with respect to a deformation of Ω_{int}. In general, the coefficient for the regularization term is set to higher values when edge data are insufficient for the determination of the object contour. In the experiments shown in the next subsection, we use $\beta = 4.0 \times 10^{-4}$ for the sequences in Figs. 9 and 10. In Figs. 7 and 8, we use $\beta = 0$.

B. Results

We have tested the proposed method for several video clips. Results are shown in Figs. 7–10.

In case of table tennis, the ball moves fast while the camera pans. When the ball hits the table, it shows an abrupt changes in motion direction. The algorithm can follow these changes as is seen in Fig. 7(c).

Fig. 8 shows another example where the algorithm demonstrates its capability of tracking a fast moving object. The object contour is detected well as it is seen with a high contrast against the background. The object has sharp limbs and as the algorithm poses no severe smoothness constraint, the limbs are detected correctly.

In the sequence of Fig. 9, the camera pans to the right, tracking a body of a man walking. In the middle of the sequence, two children enter the scene and run in opposite direction of the man. Tracking is difficult as the nonrigid object exhibits complex motion and moves in a cluttered background. Furthermore, in several frames, the background has a similar grey intensity as the head so that the head border is left undetected. This could be mended by selecting another edge detector or another tuning of the edge detector. Here, we rather demonstrate that even in the case of missing edges the

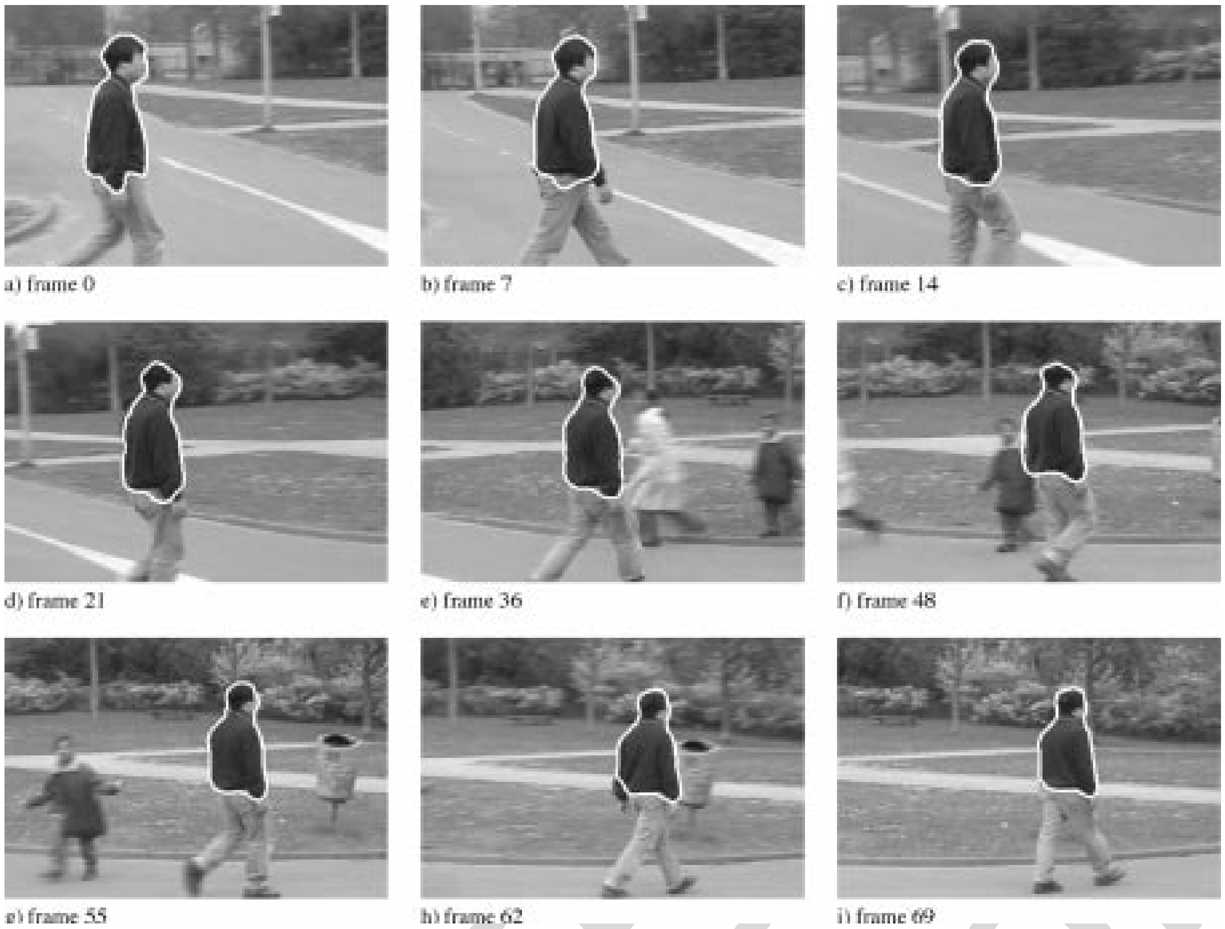

Fig. 9. Tracking results.

algorithm produces approximate results, see Fig. 9(d)–(f). The real contour is recaptured later on in the sequence as is seen in Fig. 9(e). The example also illustrates the insensitivity of the algorithm to the presence of multiple moving objects.

Another difficult situation is shown in Fig. 10(e). The motion pattern in this sequence is a rotation of the head due to which a totally different view of the object appears, followed by a move into the distance. The camera is panning slightly. When the head is rotating, part of the contour at the left ear is split into two edges: an edge between the dark hair and the ear, and another edge between the hair and the background. If no regularization is performed, the resulting contour would stick to the first edge that is undesired. To resolve this problem, edge data are insufficient. *A priori* knowledge regarding the object shape should be used. In the example shown, the regularization of (18) prevented sharp corners which usually appear in irregular shapes.

From Fig. 10(e)–(h) it is seen how the algorithm recovers from erroneous results and recaptures the real contour.

VI. Discussion

We discuss below the achievements and remaining problems with respect to the requirements put forward in the introduction.

1) *Ability to track under the condition of a moving camera and the presence of multiple motion*

This demand is satisfied, since we do not make any assumption on the motion of the camera. The camera is moving in all examples shown. Further, the presence of other moving objects in the scene does not affect the performance of the algorithm. This is an advantage over the method of [12].

2) *Ability to track fast moving objects and abrupt motion changes*

Since the object motion is estimated by an exhaustive search in the space of translation vectors, the method is able to track fast moving objects. This is an advantageous feature over many methods in literature. Note also that the algorithm is able to track abrupt changes in motion direction.

3) *Ability to track nonrigid objects*

The method shows a stable performance with no restrictions on camera motion as in [12], when the magnitude of nonrigid motion of the object is small and the contour prediction is accurate enough. The algorithm is less stable when the object motion is highly nonrigid, or complicated to the degree that the estimated dominant motion is very inaccurate. This usually happens with limbed objects whose limbs obey different motions.

4) *Robustness to missing edge information*

The combination of the current edge map and the predicted contour yields robustness to a short-time lack of edge data in the contour. In Fig. 9 specifically, the algorithm succeeds in recovering from absent edge information even when it is gone for a long time.

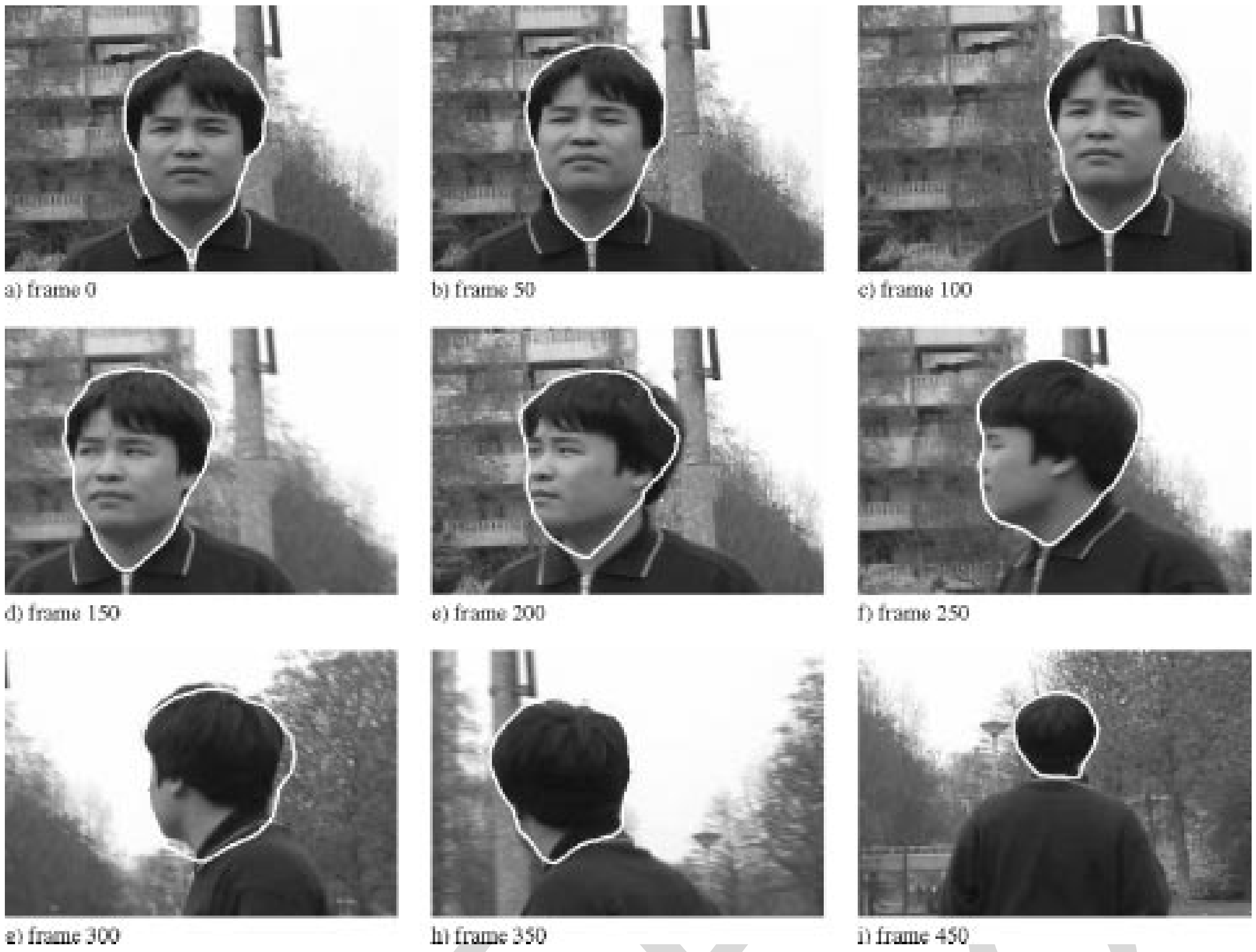

Fig. 10. Tracking results.

5) *Robustness to image clutter*

The robustness to clutter is achieved due to two factors: an accurate contour prediction and the removal of background edges. We illustrate this in Fig. 11. As observed, although the initial contour sticks to a strong spurious edge in the background, it is forced to move out from this trap in the next frames due to the contour prediction. The removal of background edges also speeds up the convergence to the real object contour.

While the background clutter can be reduced by the compensation with the object motion, the clutter inside object remains a difficult problem since the interior edges have the same motion as the contour. Another problem is that the desired contour is not always associated with the same edge during the sequence. As the object changes pose, it may happen that the contour is split into different edges.

Despite robustness against missing edges and image clutter, in general the proposed algorithm is not robust against occlusions. The reason is that the algorithm needs not only edge information for finding the object contour, but also intensity information for estimating object motion. When the object is occluded, the motion estimation will fail due to lack of data.

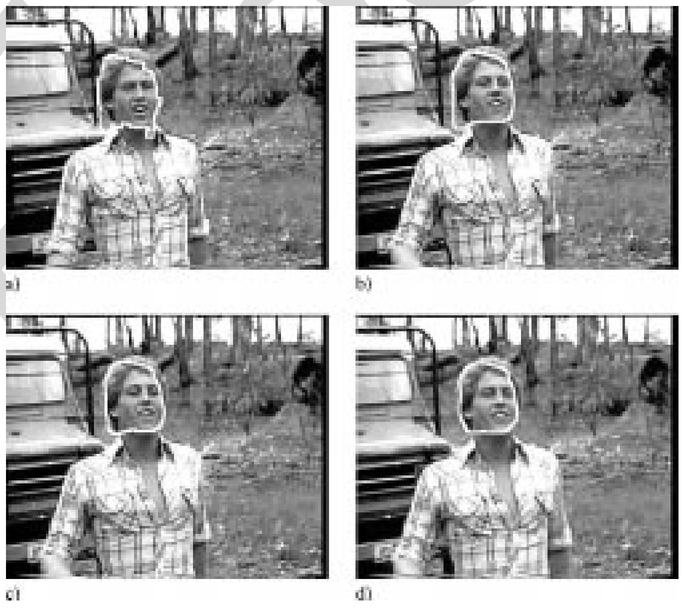

Fig. 11. Illustration of the effect of removing background edges. (a) Initial frame 0. Note that the contour is trapped at a strong spurious edge. (b) Result at frame 3 without the removal of background edges. Note that the contour prediction pulls the contour out of the trap. (c) Results at frame 3 when background edges are removed as in Section III-D. Note the improvement in comparison with (b). (d) Results at frame 5 using background edge compensation.

VII. Conclusion and Future Work

In this paper, a new method has been developed for tracking a nonparameterized contour in video. Using the inverted distance transform of the edge map as edge indicator function, the contour is prevented from being attracted to spurious edges with high gradient magnitude. The estimation of the dominant object motion allows to track fast moving objects. Contour prediction and removal of spurious edges in background by compensation with the object motion, yield robustness to background clutter. Using the concept of topographical distance, we have developed a method for combining the results of the watershed algorithm on different surfaces. This allows to increase tracking stability by taking closeness to both the current edge map and the predicted contour into account for the detection of the object contour. As the contour detection is done via energy minimization, *a priori* knowledge about the object shape can be incorporated when needed.

The algorithm can be improved in several ways. We would like to improve the edge indicator function $h(\mathbf{x})$ in such a way that it is not affected by spurious strong edges, and at the same time, takes the strength of weak edges into account. Concerning the motion estimation, one can apply the Kalman filter to smooth object's trajectory. The filter yields a better estimate of the motion vector, when it is known beforehand that the object motion is smooth. As noted in Section III-D, the compensation with object motion will not remove the edge segments, which are parallel to $\mathbf{v}_p(t)$ and longer than the length of $\mathbf{v}_p(t)$. This property can be exploited for a further removal of those edges, although this also removes some object edges with the same property. Finally, we point out some directions to tackle the problem of occlusions. To handle partial occlusions one can employ the robust statistic [25] in the estimation of object motion. To cope with larger occlusions one needs a template to store the object appearance. Occlusions can be detected when the object appearance at the current frame is too different from the previous frame. The template is also used as a memory for recapturing the object when it comes out from the occlusion [26].

In general, the algorithm exhibits good performance when the object is not cluttered and the nonrigidity in object motion is not too high. When the object is cluttered with a complex motion, using pure edge data seems not sufficient. In this case, other information such as edge motion or *a priori* knowledge regarding the object shape should be incorporated.

References

[1] A. Blake, R. Curwen, and A. Zisserman, "A framework for spatio–temporal control in the tracking of visual contour," *Int. J. Comput. Vis.*, vol. 11, no. 2, pp. 127–145, 1993.

[2] M. Isard and A. Blake, "CONDENSATION—Conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, 1998.

[3] M. Worring, A. W. M. Smeulders, L. H. Staib, and J. S. Duncan, "Parameterized feasible boundaries in gradient vector-fields," *Comput. Vis. Image Understand*, vol. 63, no. 1, pp. 135–144, 1996.

[4] D. Terzopoulos and R. Szeliski, "Tracking with Kalman snakes," in *Active Vision*, A. Blake and A. Yuille, Eds. Cambridge, MA: MIT Press, 1992, pp. 3–20.

[5] N. Peterfreund, "Robust tracking of position and velocity with Kalman snakes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, pp. 564–569, June 1999.

[6] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1987.

[7] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. New York: Wiley, 1992.

[8] J. S. Liu and R. Chen, "Sequential Monte-Carlo methods for dynamic systems," *J. Amer. Statist. Assoc.*, vol. 93, pp. 1031–1041, 1998.

[9] S. J. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Halmiton–Jacobi formulations," *J. Comput. Phys.*, vol. 72, pp. 12–49, 1988.

[10] T. McInerney and D. Terzopoulos, "Topologically adaptable snakes," in *Proc. IEEE Conf. Computer Vision, ICCV95*, 1995, pp. 840–845.

[11] K. F. Lai and R. T. Chin, "Deformable contours: Modeling and extraction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 1084–1090, Nov. 1995.

[12] N. K. Paragios and R. Deriche, "A PDE-based level set approach for detection and tracking of moving objects," in *Proc. Int. Conf. Computer Vision*, 1998, pp. 1139–1145.

[13] Ch. Gu, "Multivalued morphology and segmentation-based coding," Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 1995.

[14] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Int. J. Comput. Vis.*, vol. 22, no. 1, pp. 61–79, 1997.

[15] H. T. Nguyen, M. Worring, and A. Dev, "Detection of moving objects in video using a robust motion similarity measure," *IEEE Trans. Image Processing*, vol. 9, pp. 137–141, Jan. 2000.

[16] J. Serra, *Image Analysis and Mathematical Morphology*. New York: Academic, 1982.

[17] S. Beucher and F. Meyer, "The morphological approach of segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*, E. Dougherty, Ed. New York: Marcel Dekker, 1992, ch. 12, pp. 433–481.

[18] D. L. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2D and 3D images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1131–1147, 1993.

[19] G. Borgefors, "Distance transforms in digital images," *Comput. Vis., Graph., Image Process.*, vol. 34, pp. 344–371, 1986.

[20] F. Meyer and S. Beucher, "Morphological segmentation," *J. Vis. Commun. Image Represent.*, vol. 1, no. 1, pp. 21–46, Sept. 1990.

[21] H. T. Nguyen, M. Worring, and R. van den Boomgaard. (2000) Watersnakes: Energy-driven watershed segmentation. Intelligent Sensory Information Systems Group, Univ. of Amsterdam. [Online]. Available: http://www.science.uva.nl/research/reports-isis/2000/ISISreport12.ps.

[22] L. Vincent, "Morphological gray scale reconstruction in image analysis: Applications and efficient algorithms," *IEEE Trans. Image Processing*, vol. 2, pp. 176–201, 1993.

[23] F. Meyer, "Topographic distance and watershed lines," *Signal Process.*, vol. 38, no. 1, pp. 113–125, July 1994.

[24] L. Najman and M. Schmitt, "Watershed for a continuous function," *Signal Process.*, vol. 38, no. 1, pp. 99–112, Jul. 1994.

[25] S. Ayer and H. S. Sawhney, "Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding," in *ICCV '95*, Cambridge, 1995, pp. 777–784.

[26] H. T. Nguyen, M. Worring, and R. van den Boomgaard, "Occlusion robust adaptive template tracking," in *Proc. IEEE Conf. Computer Vision, ICCV'2001*, 2001, pp. I: 678–683.

Hieu Tat Nguyen was born in Hanoi, Vietnam, in 1971. He received the Eng. and M.Sc. degrees in computer technology from the University Lvivska Polytechnica, Lviv, The Ukraine, in 1994. He received the Ph.D. degree in computer science from the University of Amsterdam, Amsterdam, The Netherlands, in 2001.

He is currently a Postdoctoral Fellow with the Intelligent Sensory Information Systems Group at the University of Amsterdam. His current research interest includes image sequence analysis, object tracking, object recognition, active learning, mathematical morphology, and content-based image retrieval.

Marcel Worring received the M.S.(Hons.) degree in computer science from the Free University of Amsterdam, Amsterdam, The Netherlands. He received the Ph.D. degree from the University of Amsterdam in 1993. His dissertation was on digital image analysis.

He became an Assistant Professor in 1995 in the Intelligent Sensory Information Systems Group at the University of Amsterdam. His current interests are in multimedia information analysis in particular document and video analysis. He has been a Visiting Researcher in the Department of Diagnostic Imaging at Yale University, New Haven, CT, (1992), and at the Visual Computing Lab at the University of California, San Diego (1998).

Rein van den Boomgaard graduated from Delft University, Delft, The Netherlands, in 1988 and received the Ph.D. degree from the University of Amsterdam, Amsterdam, The Netherlands, in 1992.

He is an Assistant Professor with the Intelligent Sensory Information Systems (ISIS) Group at the University of Amsterdam. He is working in the field of computer vision and image processing with research interests in color vision, scale-space theory, and mathematical morphology. Since 1999 he is also an independent consultant for companies needing advice and feasibility studies in the areas of image processing and computer vision.

Arnold W. M. Smeulders received the M.Sc. degree in physics from the Technical University of Delft, Delft, The Netherlands, in 1977, and the Ph.D. degree in medicine (on the topic of visual pattern analysis) from Leyden University, The Netherlands.

He heads the ISIS Research Group of 15 Ph.D. students and ten staff members concentrating on theory, practice, and implementation of multimedia information analysis including image databases and computer vision. His current research is in multimedia information analysis and computer vision of the real world. His focus is covered by the ICPR, CVPR, ICME, and Visual Conference Series. He is a board member of the International Association of Pattern Recognition.

Dr. Smeulders was elected Fellow of International Association of Pattern Recognition. He was associate editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.