Active Contour-Based Visual Tracking by Integrating Colors, Shapes, and Motions

Weiming Hu, Xue Zhou, Wei Li, Wenhan Luo, Xiaoqin Zhang, and Stephen Maybank

***Abstract*— In this paper, we present a framework for active contour-based visual tracking using level sets. The main components of our framework include contour-based tracking initialization, color-based contour evolution, adaptive shape-based contour evolution for non-periodic motions, dynamic shape-based contour evolution for periodic motions, and the handling of abrupt motions. For the initialization of contour-based tracking, we develop an optical flow-based algorithm for automatically initializing contours at the first frame. For the color-based contour evolution, Markov random field theory is used to measure correlations between values of neighboring pixels for posterior probability estimation. For adaptive shape-based contour evolution, the global shape information and the local color information are combined to hierarchically evolve the contour, and a flexible shape updating model is constructed. For the dynamic shape-based contour evolution, a shape mode transition matrix is learnt to characterize the temporal correlations of object shapes. For the handling of abrupt motions, particle swarm optimization is adopted to capture the global motion which is applied to the contour in the current frame to produce an initial contour in the next frame.**

***Index Terms*— Abrupt motion, active contour-based tracking, adaptive shape model, dynamic shape model.**

I. Introduction

VISUAL object tracking is an active research topic in computer vision. In contrast to general object tracking which uses predefined coarse shape models, such as rectangles or ellipses, to represent objects [5], active contour-based tracking [36] provides more detailed object shape information, but is, in general, more difficult than general tracking of the same object in the same real-world situation. This is because contour tracking aims to recover finer details of the object, i.e., the boundary of the object, and the determination of the boundary of the object is susceptible to influences from the background disturbance. In videos taken by stationary cameras, object motion regions can often be extracted using background subtraction, and object contours can be produced by tracing the edges of the motion regions. But in videos taken by moving cameras, background subtraction cannot be used to extract object motion regions, making the contour-based tracking more difficult than in videos taken by stationary cameras. Active contour-based object tracking, no matter whether the camera is stationary or moving, has attracted much attention in recent years.

Manuscript received September 2, 2011; revised April 17, 2012; accepted July 19, 2012. Date of publication December 24, 2012; date of current version March 11, 2013. This work was supported in part by the NSFC under Grant 60825204, Grant 60935002, Grant 60905015, and Grant 61100147, the National 863 High-Tech R&D Program of China under Grant 2012AA012504, the Natural Science Foundation of Beijing under Grant 4121003, and the Guangdong Natural Science Foundation through a project under Grant S2012020011081. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ying Wu.

W. Hu, W. Li, W. Luo, and X. Zhang are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: wmhu@nlpr.ia.ac.cn; weili nlpr.ia.ac.cn; whluo nlpr.ia.ac.cn; xqzhang@nlpr.ia.ac.cn).

X. Zhou is with the School of Automation, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: zhouxue@uestc.edu.cn).

S. Maybank is with the Department of Computer Science and Information Systems, Birkbeck College, London WC1E 7HX, U.K. (e-mail: sjmaybank@dcs.bbk.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2012.2236340

A. Related Work

There are in general two ways to describe object contours: explicit representations which are characterized by parameterized curves such as snakes [1] and implicit representations, such as level sets [3], which represent a contour using a signed distance map. The level set representation is more popular than the explicit representation because it has a stable numerical solution and it is capable of handling topological changes. Active contour evolution methods are classified into three categories: edge-based, region-based, and shape prior-based.

1) Edge-Based Methods: Edge-based methods mainly consider the local information around contours, such as the grey level gradient. Kass *et al.* [1] propose the snake model which is the best known edge-based active contour method. Caselles *et al.* [12] propose a geodesic model which reflects more intrinsic geometric image measures than the snake, using the prior knowledge that the larger the gradient at a pixel, the higher the probability that the pixel belongs to an object's edge. Paragios and Deriche [13] improve the geodesic model in [12] using level sets to describe contours and using a gradient descent algorithm to optimize contours. The merits of edge-based methods are their simplicity, intuitiveness, and effectiveness for determining contours with salient gradient. They have the following limitations: a) They only consider the local information near to a contour, and thus the initial contour must be near to the object. b) Contour sections lying in homogeneous regions of an image cannot be optimized. c) They are of course sensitive to image noise.

2) Region-Based Methods: Region-based methods usually divide an image into object and background regions using

1057–7149/$31.00 © 2012 IEEE

statistical quantities, such as mean, variance, or histograms of the pixel values in each region. Chan and Vese [9] approximate an image by a mean image with regions whose boundaries are treated as object edges. Zhu and Yuille [15] present a statistical and variational framework for image segmentation using a region competition algorithm. Yilmaz *et al.* [16] adopt the features of both object and background regions in the level-set evolution model. Mansouri [32] proposes an algorithm for formulating contour tracking as a Bayesian estimation problem. For the region-based methods, prior knowledge of object color and texture may be incorporated into the contour evolution process. Color prior knowledge is usually represented using object appearance models such as color histograms, kernel density estimation, or Gaussian mixture models (GMMs). For example, Yilmaz *et al.* [16] use kernel density estimation to model color features for estimating contours. Bibby and Reid [43] use color histograms to model appearances and perform contour-based tracking at frame rates. Region texture features [41], [42] are usually modeled using the Gabor filer, the co-occurrence matrix, or Markov random fields (MRF), etc. For example, Sagiv *et al.* [41] use Gabor features to perform textured image segmentation. Pons *et al.* [42] fit an active contour using texture features which are extracted using unsupervised learning. Yilmaz *et al.* [16] use the Gabor filter to model region texture features for determining contours. The merit of the region-based methods is that regions' statistical information, together with the prior knowledge of object color and texture, can increase the robustness and accuracy of contour evolution. The limitation of the current region-based methods is that the pixel values are treated as if they were independent for the posterior probability estimation [36]. This independence assumption makes the obtained contour sensitive to disturbances caused by similarities of color or texture between the object and the background.

3) Shape Prior-Based Methods: Shape prior-based methods statistically model object shape priors which are used to recover disturbed, occluded, or blurred contour sections. Leventon *et al.* [18] project orthogonally a set of aligned training shape samples represented by the signed distance maps into a subspace using Principal Component Analysis (PCA). Paragios and Rousson [20] construct a pixel-wise shape model in which local shape variability can be accounted for. Cootes *et al.* [21] propose an active shape model for the different aspects of rigid objects in a shape prior formulation. Fussenegger *et al.* [22] propose an online active shape model to perform region segmentation. The incremental PCA algorithm in [23] is used to update the active shape model. Cremers [19] proposes a linear dynamical shape model based on an autoregressive model for tracking a person with periodic motions using level sets. Yilmaz *et al.* [28] propose a statistical method to learn object shape models which are used to recover occluded sections of a contour. Rathi *et al.* [33] combine the particle filter with level set evolution. Occlusion is dealt with by incorporating shape information into the weights of the particles. Raviv *et al.* [38] utilize the symmetry of rigid object shapes to deal with partial occlusions. The merit of the shape prior-based methods is that the disturbed, occluded, or blurred edges can be recovered. However, the current adaptive shape-based methods [20] may distort undisturbed contour sections which can be found accurately using color features alone, while they globally recover the disturbed contour sections. In real world applications it is necessary to update the active shape model continuously in order to adapt to shape changes. However, the current method [22] for updating the shape model does not simultaneously handle the multiple new shape samples, and fails to compute the sample eigenbasis with sample mean updating. The previous dynamical shape model in [19] for periodic motions of non-rigid objects is a simple data fitting process with no high-level understanding of shape changes. The model assumes that the underlying motion is closely approximated by a periodic motion, however human motion is rarely exactly periodic.

Current contour-based tracking algorithms are subject to additional limitations as follows. 1) Tracking initialization often relies on a manually drawn closed contour around the object. Those methods, in which the boundaries of motion regions detected by background subtraction are the initial contours of moving objects such as in [13], are only effective for stationary cameras. In [9], the initial contour can be placed anywhere in the image, but it may take a long time to converge to the correct boundary. Quick and automatic initialization of contour tracking is still underdeveloped and demanding. 2) The existing level set-based tracking methods fail to track the contour of an object when the object moves abruptly. Related work in [34], [35] deals with the discontinuities induced by abrupt motion. However, the robust and effective handling of abrupt motion for contour-based tracking is still a difficult open problem.

B. Our Work

In this paper, we systematically investigate the aforesaid main limitations in contour tracking, and present a framework for tracking object contours, no matter whether the camera is stationary or moving. The framework adopts the region-based evolution of contours which are represented using level sets. At the first frame, the method in [40] is used to compensate for the camera motion and then the optical flow at each pixel is estimated. Using the estimated optical flows, one or more motion regions are detected. The boundaries of these motion regions are used as the initial contours. These initial contours are then evolved using color information. Based on the result of color-based contour evolution, the shape prior is introduced to deal with noise or partial occlusion etc to obtain more accurate contours. We consider shape priors for non-periodic motions and periodic motions, corresponding to adaptive shape models and dynamic shape models respectively. After contour evolution is complete in the current frame, there is a check for abrupt motion which can be estimated using the method in [39]. If there is no abrupt motion, the evolution result in the current frame is used as the initial contour of the object in the next frame. If there is abrupt motion, the affine motion parameters are estimated using a stochastic algorithm, and applied to the contour in the current frame to obtain the initial contour for the next frame. The main components in our framework include contour-based tracking initialization

for the first frame, color-based contour evolution, adaptive shape-based contour evolution, dynamic shape-based contour evolution, and abrupt motion handling.

The main components in our framework have the following contributions:

1) We propose an automatic and fast tracking initialization algorithm based on optical flow detection. In the algorithm, object motion regions are extracted in the first frame, and closed initial contours near the boundaries of object regions are constructed.
2) We propose a color-based contour evolution algorithm. In the algorithm, correlations between values of neighboring pixels are constructed using Markov random field (MRF) theory, and incorporated into the estimation of the posterior probability of segmentation. This ensures that our color-based algorithm is not sensitive to background disturbances and that it achieves tight and smooth contours.
3) We propose an adaptive shape-based contour evolution algorithm. In the algorithm, the results obtained using the color feature alone and the shape priors are effectively combined, adapting to different contour locations, to obtain the final contour. A new incremental PCA technique is applied to update the shape model, making the shape model updating flexible.
4) We propose a Markov model-based dynamical shape model. Dominant set clustering is used to obtain the typical shape modes of a periodic motion. The matrix of transitions between these modes is then constructed. In the tracking process, the contour evolved using color information alone and the shape mode transition matrix are fused to predict the current shape mode, and then the contour is further evolved under the constraint of the predicted shape mode.
5) We propose an algorithm for handling abrupt motion in the contour tracking process by incorporating particle swarm optimization (PSO) into the level set evolution. Although the algorithm in [33] combines the particle filter with the level set evolution, it cannot handle abrupt and arbitrary motions. In our algorithm, PSO is used to estimate the global motion of the object. The initial contour is obtained by applying the estimated global motion to the contour of the object in the previous frame.

II. Concepts of Level Set Evolution

The level set function [3] chosen in our method is the commonly used signed distance function. The zero value of this function corresponds to a contour. The shape information of the contour C is embedded in the signed distance map represented by ϕ:

$$\phi(x,y)=\begin{cases}0 & (x,y)\in C\\ d(x,y,C) & (x,y)\in R_{\text{out}}\\ -d(x,y,C) & (x,y)\in R_{\text{in}}\end{cases} \tag{1}$$

where R_{in} and R_{out} denote, respectively, the regions inside and outside C and $d(x,y,C)$ is the smallest Euclidean distance from point (x,y) to C. The initial contour is evolved to the desired boundary by updating ϕ iteratively with the overall speed in the normal direction [8]:

$$\frac{\phi^{n+1}(x,y)-\phi^{n}(x,y)}{\Delta t}+(F_{(x,y)}+F_{\text{curv}})\,|\nabla\phi(x,y)|=0 \tag{2}$$

where $F_{(x,y)}$ is the external force reflecting the data attachment, $F_{\text{curv}}=-\varepsilon\kappa(x,y)$ is the internal force proportional to the curvature $\kappa(x,y)$ of the contour and has a smoothing effect on the contour, n represents the n-th iteration, $\nabla\phi(x,y)$ is estimated as the gradient of the level set function at (x,y), and Δt is the evolution step which can be set to a fixed value such as 0.001. The key problems in level set contour evolution include the estimation of $F_{(x,y)}$ as well as the initialization of the contour.

III. Tracking Initialization

The tracking initialization in our framework consists of localization of initial contours in the first frame and modeling of the object and background regions using color and texture features.

A. Contour Initialization in the First Frame

In the first frame, we apply optical flow to detect motion regions whose boundaries are used as the initial contours. The algorithm in [40] for ego-motion compensation is used to compensate camera motion. We combine the optical flow vector magnitude detected using Horn and Schunck's algorithm [4] and the direction detected using Lucas and Kanade's algorithm [11] to obtain the optical flow vector at each pixel, because from many experiments, we have found that Horn and Schunck's algorithm obtains more accurate magnitudes of optical flow vectors than Lucas and Kanade's algorithm, but obtains less accurate directions of optical flow vectors.

The optical flow for each pixel is represented by (u,v), where u and v are the optical flow velocity vector's components in the x and y directions respectively. For a pixel whose optical flow magnitude is less than a predefined threshold, its optical flow is set to (0, 0), i.e. it is assigned to the background. Then, a shape model such as a circle or a rectangle is moved over the image and its size is changed, to detect the motion regions in which pixels have not only large optical flow magnitudes, but also coherent optical flow directions. The detection algorithm includes the following steps:

Step 1: The size of the shape model is kept fixed and the center of the shape model is moved. Then, a series of regions $\{\mathrm{M}_i\}$, which are various sets of pixels within the shape model, are produced. Each such region M_i is assigned a weight ξ_i calculated by:

$$\xi_i=\beta\frac{\sum_{\mathrm{X}\in\mathrm{M}_i}\sqrt{u_{\mathrm{X}}^2+v_{\mathrm{X}}^2}}{\Upsilon_i}-(1-\beta)\Omega_{\arg(u,v)} \tag{3}$$

where X is a pixel within M_i, $\Omega^2_{\arg(u,v)}$ is the variance of the directions of the flow vectors of the pixels within M_i, Υ_i is the number of pixels within M_i, and β is the weight (ranging

between 0 and 1) which balances flow vector magnitude which is indicated by the first part of the right hand side of the equation and motion direction coherence which is represented by $\Omega_{\arg(u,v)}$. The greater the magnitudes of the flow vectors and the more the motion direction coherence, the larger the weight is.

Step 2: The top $N(N \geq 1)$ such regions with the largest weights $\{\xi_i\}_{i=1,2,\ldots,N}$ are chosen as the motion regions with optimal positions. The number N of these regions is determined based on the principle that the weights $\{\xi_i\}_{i=1,2,\ldots,N}$ of these regions are far larger than the weights of the others.

Step 3: We refine each of these regions by changing its size while keeping its center position fixed. The dimensions v, which control the size of the shape model, such as the radius for a circle or the length and width of a rectangle, determine the Υ in (3), so ξ is a function of v. The optimal dimension v^* of each motion region is determined by:

$$v^* = \arg\max_{v}(\xi(v)). \tag{4}$$

Step 4: The moving objects correspond to the regions with largest values of Eq. (3). The edges of shape models (circles or rectangles) with the optimal centers and sizes are treated as initial object contours. We compute the level set functions $\phi(x, y)$ of the initial contours from which the object contours are further evolved using color-based contour evolution.

B. Modeling the Object and Background Regions

In our active contour-based object tracking algorithm, the object and background regions are both modeled, and both regions compete for pixels in the image. We present a hierarchical method which fuses color and texture features using a Gaussian mixture model (GMM) which is a variant of Stauffer and Grimson's method [7], to model the object and background regions. The method is outlined as follows:

Step 1: Within the object or background region, a color GMM model with k Gaussians $\{\eta(\mathrm{X}^C, \mu_j^C, \Sigma_j^C)\}_{j=1,2,\ldots,k}$ is trained, where μ_j^C and Σ_j^C are the mean and the covariance of the jth Gaussian probability density function and X^C is the color feature of pixel X.

Step 2: Each pixel within the region is labeled based on the trained color GMM model, where the label of each pixel is the label of the Gaussian which has the maximum value for this pixel among all the Gaussians in the mixture.

Step 3: The texture features for all the pixels in the region are estimated using a gray level co-occurrence matrix method [6]. The texture features for all the pixels with the same label are modeled as a Gaussian $\eta(\mathrm{X}^T, \mu^T, \Sigma^T)$ where μ^T and Σ^T are the mean and covariance of the Gaussian and X^T is the texture feature of pixel X.

Step 4: The estimated probability density function at pixel X in the joint color-texture space is formulated as:

$$p\left(\mathrm{X}|\omega, \mu^C, \Sigma^C, \mu^T, \Sigma^T\right) = \sum_{j=1}^{k} \omega_j \eta\left(\mathrm{X}^C, \mu_j^C, \Sigma_j^C\right) \eta\left(\mathrm{X}^T, \mu_j^T, \Sigma_j^T\right) \tag{5}$$

where $\{\omega_j\}_{j=1,2,\ldots,k}$ are the weight parameters of the GMM model. The method for updating the GMM parameters is similar to that in [7].

IV. Color-Based Contour Evolution

The task in contour evolution is to adjust an initial contour until the image is partitioned optimally by the contour [15], [16] into a foreground region and a background region. Let $\Re(I)$ represent a partition of image I. In accordance with Bayes' rule, the posterior probability $P(\Re(I)|I)$ can be represented as: $P(\Re(I)|I) \propto P(I|\Re(I))P(\Re(I))$. The prior probability $P(\Re(I))$ encodes a motion/dynamical model or shape information for the region. In various implementations, $P(\Re(I))$ acts as a smoothing regularization term which depends on the length of the contour [14], [17]. Since the aim of the partitioning is to separate image regions whose properties are different, it is assumed that certain statistical properties of the foreground and background regions are independent. According to [36], for posterior probability and likelihood estimation, current methods assume that the pixel values in the object region or the background region are independent [15], [16]. Then, the following equation is obtained:

$$P(I|\Re(I)) = \prod_{\mathrm{X}_i \in R_{\text{in}}} P(v(\mathrm{X}_i)|\mathrm{X}_i \in R_{\text{in}}) \prod_{\mathrm{X}_j \in R_{\text{out}}} \times P(v(\mathrm{X}_j)|\mathrm{X}_j \in R_{\text{out}}) \tag{6}$$

where R_{in} and R_{out} denote the regions inside and outside the contour respectively corresponding to the partition $\Re(I)$, and $v(\mathrm{X})$ represents the value of pixel X. The terms $P(v(\mathrm{X})|\mathrm{X} \in R_{\text{in}})$ and $P(v(\mathrm{X})|\mathrm{X} \in R_{\text{out}})$ are estimated using the color and texture distributions of the object and background regions respectively. However, the hypothesis of independence of pixel values for posterior probability and likelihood estimation is too strong, especially when there are local associations between pixels, such as for textured regions or regions with repeated patterns [36]. As a result, it is easy to misidentify pixels around object boundary sections where the contrast between the object and the background is low.

To avoid the above problem, we apply the Markov random field (MRF) theory to take account of the correlations between the values of neighboring pixels for posterior probability and likelihood estimation. According to the MRF theory, the value of a pixel is only correlated with the values of the pixels in its neighborhood [10] which is assumed to be a square pixel region centered at it, i.e., its value is independent of the values of the pixels outside the neighborhood. Then, the posterior probability can be written as:

$$P(\Re(I)|I) = \prod_{\mathrm{X} \in I} P(\Re(\mathrm{X})|v(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) \propto \prod_{\mathrm{X} \in I} P(v(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\Re(\mathrm{X}))P(\Re(\mathrm{X})) \tag{7}$$

where $\mathbb{N}_{\mathrm{X}}$ means the value of the neighborhood pixels of X. For the prior probability $P(\Re(\mathrm{X}))$, it is assumed that

$$P(\Re(\mathrm{X})) = P(\mathrm{X} \in R_{\text{in}}) = P(\mathrm{X} \in R_{\text{out}}). \tag{8}$$

Formula (7) is rewritten as:

$$P(\Re(I)|I) \propto \prod_{\mathrm{X}_i \in R_{\mathrm{in}}} P(\upsilon(\mathrm{X}_i), \mathbb{N}_{\mathrm{X}_i}|\mathrm{X}_i \in R_{\mathrm{in}}) P(\mathrm{X}_i \in R_{\mathrm{in}}) \times \prod_{\mathrm{X}_j \in R_{\mathrm{out}}} P(\upsilon(\mathrm{X}_j), \mathbb{N}_{\mathrm{X}_j}|\mathrm{X}_j \in R_{\mathrm{out}}) P(\mathrm{X}_j \in R_{\mathrm{out}}). \quad (9)$$

It is obvious that

$$P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}}) P(\mathrm{X} \in R_{\mathrm{in}}) \propto P(\mathrm{X} \in R_{\mathrm{in}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) \quad (10)$$

$$P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{out}}) P(\mathrm{X} \in R_{\mathrm{out}}) \propto P(\mathrm{X} \in R_{\mathrm{out}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}). \quad (11)$$

Given the value of a pixel and the values of the pixels in its neighborhood, its object or background attribute is determined. This leads to $P(\mathrm{X} \in R_{\mathrm{in}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) + P(\mathrm{X} \in R_{\mathrm{out}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) = 1$. Using Bayes' rule, the following equations are obtained:

$$\begin{aligned} &P(\mathrm{X} \in R_{\mathrm{in}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) \\ &= \frac{P(\mathrm{X} \in R_{\mathrm{in}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}})}{P(\mathrm{X} \in R_{\mathrm{in}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) + P(\mathrm{X} \in R_{\mathrm{out}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}})} \\ &= (P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}}) P(\mathrm{X} \in R_{\mathrm{in}}))/ \\ &\quad (P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}}) P(\mathrm{X} \in R_{\mathrm{in}}) \\ &\quad + P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{out}}) P(\mathrm{X} \in R_{\mathrm{out}})) \\ &= \frac{P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}})}{P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}}) + P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{out}})} \end{aligned} \quad (12)$$

$$P(\mathrm{X} \in R_{\mathrm{out}}|\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}) = \frac{P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{out}})}{P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}}) + P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{out}})}. \quad (13)$$

According to the equation for conditional probability, it follows that

$$P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{in}}) = P(\upsilon(\mathrm{X})|\mathrm{X} \in R_{\mathrm{in}}) \times P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{in}}) \quad (14)$$

$$P(\upsilon(\mathrm{X}), \mathbb{N}_{\mathrm{X}}|\mathrm{X} \in R_{\mathrm{out}}) = P(\upsilon(\mathrm{X})|\mathrm{X} \in R_{\mathrm{out}}) \times P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{out}}). \quad (15)$$

The terms $P(\upsilon(\mathrm{X})|\mathrm{X} \in R_{\mathrm{in}})$ and $P(\upsilon(\mathrm{X})|\mathrm{X} \in R_{\mathrm{out}})$ can be estimated, respectively, using the parameters of the object GMM and the background GMM: $\{\omega_{\mathrm{in}}, \mu_{\mathrm{in}}^C, \sum_{\mathrm{in}}^C, \mu_{\mathrm{in}}^T, \Sigma_{\mathrm{in}}^T\}$ and $\{\omega_{\mathrm{out}}, \mu_{\mathrm{out}}^C, \sum_{\mathrm{out}}^C, \mu_{\mathrm{out}}^T, \Sigma_{\mathrm{out}}^T\}$ in (5). The terms $P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{in}})$ and $P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{out}})$ measure the correlations between the values of the neighboring pixels and the value of the center pixel. For a neighborhood centered at pixel X, let N_O be the number of the neighboring pixels which belong to the object, i.e., the pixel's probability of belonging to the object is greater than the probability that it belongs to the background; and let N_B be the number of the neighboring pixels which belong to the background. We heuristically approximate $P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{in}})$ and $P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{out}})$ by:

$$P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{in}}) \propto \exp\left(sign(N_O - N_B) * \left(\frac{\max(N_O, N_B)}{N_O + N_B}\right)^2 \Big/ \sigma^2\right) \quad (16)$$

$$P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{out}}) \propto \exp\left(sign(N_B - N_O) * \left(\frac{\max(N_O, N_B)}{N_O + N_B}\right)^2 \Big/ \sigma^2\right) \quad (17)$$

where σ is the parameter controlling how fast the exponential function converges to zero. The more N_O exceeds N_B, the larger $P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{in}})$; and the more N_B exceeds N_O, the larger $P(\mathbb{N}_{\mathrm{X}}|\upsilon(\mathrm{X}), \mathrm{X} \in R_{\mathrm{out}})$.

It is noted that selection of an appropriate size of the neighborhood is a tradeoff between accuracy and smoothness of the tracking results. The larger the size, the smoother the contour we can obtain, but the more the details of the contour are lost. By constructing correlations between the values of the neighborhood pixels as described in (16) and (17), the probability of misidentifying the background pixels whose colors are similar to the object colors is reduced, making the algorithm more robust to background disturbance.

By converting the problem of maximizing $P(\Re(I)|I)$ in (9) to an energy minimization problem, the following energy equation is obtained:

$$\begin{aligned} E &= -\log P(\Re(I)|I) \\ &\approx -\iint_{\mathrm{X}_i \in R_{\mathrm{in}}} \log P(\mathrm{X}_i \in R_{\mathrm{in}}|\upsilon(\mathrm{X}_i), \mathbb{N}_{\mathrm{X}_i}) d\mathrm{X}_i \\ &\quad - \iint_{\mathrm{X}_j \in R_{\mathrm{out}}} \log P(\mathrm{X}_j \in R_{\mathrm{out}}|\upsilon(\mathrm{X}_j), \mathbb{N}_{\mathrm{X}_j}) d\mathrm{X}_j. \end{aligned} \quad (18)$$

Let $\mathrm{X}_{(x,y)}$ represent the pixel at location(x, y). Let $H_a(\phi(x, y))$ be a Heaviside function:

$$H_a(\phi(x, y)) = \begin{cases} 0 & \phi(x, y) \geq 0 \\ 1 & \phi(x, y) < 0 \end{cases} \quad (19)$$

which takes "1" if (x, y) is inside the contour and "0" if (x, y) is on or outside the contour. By minimizing the energy function shown in (18) through solving the correlated Euler–Lagrange equations (See [16] for details), we obtain the level set evolution speed model $F_{x,y}$ in (2). It is represented in the following way when the neighborhood of each pixel is defined as a $(2l + 1) \times (2l + 1)$ square region centered at the pixel:

$$\begin{aligned} F_{x,y} = &- \sum_{i=-l}^{l} \sum_{j=-l}^{l} \log P(\mathrm{X}_{(x+i,y+j)} \in R_{\mathrm{in}}| \\ &\upsilon(\mathrm{X}_{(x+i,y+j)}), \mathbb{N}_{(x+i,y+j)}) H_a(\phi(x + i, y + j)) \\ &+ \sum_{i=-l}^{l} \sum_{j=-l}^{l} \log P(\mathrm{X}_{(x+i,y+j)} \in R_{\mathrm{out}}|\upsilon(\mathrm{X}_{(x+i,y+j)}), \\ &\mathbb{N}_{(x+i,y+j)})(1 - H_a(\phi(x + i, y + j))). \end{aligned} \quad (20)$$

The contour is evolved to the desired boundary by modifying ϕ iteratively using the equation obtained by substituting $F_{x,y}$ in (20) into Eq. (2).

V. Adaptive Shape-Based Contour Evolution

Object shape information can be used to improve the results of color-based contour evolution. We propose a hierarchical shape-based contour evolution algorithm which combines both the global shape information and the local color information to obtain a contour closer to the true contour. On the basis of shape registration and construction of a shape subspace, the Mahalanobis distance-based criterion is used to determine whether the shape model is introduced into the contour evolution process. If so, a shape-based evolution which adapts to different contour locations is used to further evolve the contour. A novel incremental PCA is applied to update the shape model in a more flexible way than in the previous shape model updating algorithm [22].

A. Shape Registration

The shape of each contour is represented using its corresponding level-sets signed distance map ϕ. Shape registration from shape A to shape B involves scaling, rotating, and translating shape A to obtain a new shape which best matches shape B. We use Paragios' variational method [26], to implement the shape registration.

B. Construction of Shape Subspace

The contour shape subspace is constructed in the following way which is similar to that of [18]. From a training sequence, we manually obtain a series of training shape samples of the object which is to be tracked in the test sequence. All the signed distance maps of each sample are aligned using the shape registration. The level set embedding function values in each distance map are flattened into a column vector [19]. The mean vector μ is obtained by taking the mean of the column vectors for all the samples. A matrix X whose columns are obtained by subtracting μ from each sample column vector is constructed. Using the singular value decomposition (SVD) for X, the diagonal matrix Σ_k with the first k largest singular values and the corresponding singular column vector matrix U_k are obtained. The shape model is represented by $\{\mu, U_k, \Sigma_k\}$.

C. Mahalanobis Distance-Based Criterion

If there is no background disturbance, motion blurring, partial occlusion, etc, then the color-based evolution alone can obtain a contour very close to the true contour in the image. The introduction of the shape prior into the contour evolution may even draw the obtained contour away from the true contour. In this paper, we use the Mahabanobis distance between the contour ϕ_c obtained using the color-based evolution alone and the shape model to determine whether the shape prior should be introduced into the contour evolution.

The contour ϕ_c is aligned with the signed distance map ϕ_T [18] corresponding to the mean shape μ in the shape subspace using the shape registration. The level set embedding function values of the aligned contour are flattened into a column vector x, and then projected into the subspace, forming a k-dimensional vector α: $\alpha = U_k^T(x - \mu)$.

The Mahalanobis distance is applied to measure the difference between x and μ. Let C be N times the covariance matrix $C \approx U_k \Sigma_k^2 U_k^T$, where N is the number of samples. The Mahalanobis distance between x and μ is formulated as: $\gamma^2 = (x - \mu)^T C^{-1}(x - \mu)$. Then, according to the definition of α, the following equation is obtained:

$$\gamma^2 \approx (x - \mu)^T U_k \Sigma_k^{-2} U_k^T (x - \mu) = \alpha^T \Sigma_k^{-2} \alpha. \tag{21}$$

If γ^2 is larger than a predefined threshold, then the color-based evolution result ϕ_c does not accurately represent the object shape information, and the shape prior is introduced into the tracking process; otherwise ϕ_c is considered as the final tracking result.

D. Shape-Based Contour Evolution

The signed distance map ϕ_T [18] corresponding to μ is inversely transformed to the image plane based on the transformation parameters for shape registration between ϕ_c and ϕ_T to form a signed distance map ϕ_s, i.e., the contour ϕ_s is obtained by scaling, rotating, and translating ϕ_T so that ϕ_s best matches ϕ_c.

The shape prior ϕ_s can be used to globally recover the contour sections which are disturbed by the background, blurred by motion, or even occluded by the background, but it may distort contour sections which have been obtained accurately using color features alone. The task of our algorithm is to fuse ϕ_c and ϕ_s to form a contour closer than either to the true contour.

We determine whether a contour section on ϕ_c is closer to the true contour section in the image than the corresponding contour section on the shape prior, according to the distance between the contour section on ϕ_c and the corresponding contour section on the shape prior, as simple and known properties of the object are included in the shape prior. A large distance means that the contour section on ϕ_c is erroneous due to background disturbance, motion blurring or occlusions, etc. A small distance indicates that the contour section obtained on ϕ_c is closer to the true contour section in the image.

In contrast with previous methods [20], [31], we add a shape distance weight into the contour evolution equation to balance ϕ_c and ϕ_s:

$$\frac{\phi^{n+1}(x, y) - \phi^n(x, y)}{\Delta t} = -2(\phi^n(x, y) - \phi_s(x, y)) \times \left[1 - \exp\left(-\left(\frac{\phi^n(x, y) - \phi_s(x, y)}{\tau(x, y)}\right)^2\right)\right] \tag{22}$$

where τ is the parameter controlling the speed that the exponential function converges to zero, and the definitions of n and Δt are the same as in (2). The initial value of ϕ is set to ϕ_c. The weight in the square bracket in (22) is a function of the difference between $\phi(x, y)$ and $\phi_s(x, y)$. It has the following characteristics:

1) For a pixel location (x, y), if the difference between $\phi(x, y)$ and $\phi_s(x, y)$ is not large, then the weight is close to zero and thus the value of $\phi(x, y)$ changes only very slightly, i.e., approximating to $\phi_c(x, y)$.

2) If the difference between $\phi(x, y)$ and $\phi_s(x, y)$ is large, then the weight is close to 1, and thus the evolution process for $\phi(x, y)$ goes on until its value approximates to $\phi_s(x, y)$.

The iterations using (22) are repeated for a given number of times or until the mean of differences between $\phi(x, y)$ and $\phi_s(x, y)$ at each location (x, y) is less than a predefined threshold.

For a pixel location (x, y) where the difference between $\phi(x, y)$ and $\phi_s(x, y)$ is large at the beginning of the iterations, if $\phi(x, y)$ converges too soon, inaccurate results are obtained. Then, we adjust the parameter τ in the following way to control the evolution speed at different pixel locations:

$$\tau(x, y) = \beta \exp(-(\phi_c(x, y) - \phi_s(x, y))^2) \tag{23}$$

where β is a positive constant. Thus, at each pixel location (x, y) where the difference between $\phi_c(x, y)$ and $\phi_s(x, y)$ is large, the evolution of $\phi(x, y)$ can still continue when the difference between $\phi(x, y)$ and $\phi_s(x, y)$ is getting smaller and smaller. At each pixel location (x, y) where the difference between $\phi_c(x, y)$ and $\phi_s(x, y)$ is small, the evolution of $\phi(x, y)$ stops soon. In this way, the result obtained using the color feature alone and the shape prior are combined effectively to bring the final contour closer to the true contour in the image.

E. Incremental Updating of the Shape Model

Online updating of the shape model using the contours obtained from the recent frames is necessary for shape-based contour tracking, as the recent frames provide more accurate information about the current shape of the object. To date, the only published algorithm for updating the shape model online is the one described in [22]. The subspace-based contour shape model in [22] uses the incremental PCA algorithm in [23] to update the shape model online. The limitations of the incremental PCA algorithm in [23] are as follows: 1) Each update includes only one new sample, rather than multi-samples, making it necessary to update the shape model at every frame. 2) The update is obtained by successive approximations, and it is not accurate enough for applications.

We apply the incremental PCA algorithm in [24] to learn the changes in the shape of the object during tracking. The result of incremental PCA in [24] for a matrix is the same as the result of the PCA for the matrix in batch mode, provided that the incremental PCA retains all the singular values and singular vectors at each step. If the singular vectors associated with the smaller singular values are discarded, then the algorithm for the incremental PCA becomes faster, with little loss of accuracy. Hence, the subspace learned using this incremental PCA is very accurate. Furthermore, this incremental PCA updates the subspace using a number of new samples simultaneously. It includes an update of the mean, removing the assumption [25] that the mean of the previous data is equal to the mean of the new data. When the contour shape changes determined using the Mahalanobis distance-based criterion are small in a number of consecutive frames, the shape model is updated once using the contour evolution results in these frames. When the change is large, the shape model is updated more frequently, as much as once a frame. In this way, we obtain not only more accurate but also more flexible and more rapid shape updating.

VI. Dynamic Shape-Based Contour Evolution

Dynamic shape models are more appropriate than the adaptive shape models to deal with large changes in shapes of non-rigid object contours. In the following, we cluster shape samples of non-rigid object contours to obtain the typical shape modes and the mode transition matrix which is used to predict the shape mode. With the constraint of the predicted shape mode, the contour obtained by the color-based evolution is further evolved to obtain the final contour.

A. Construction of Shape Models

1) Distances Between Samples: The shape information about each training contour is contained in the signed distance map ϕ. We define the distance between shape samples ϕ_i and ϕ_j which are aligned using shape registration by:

$$d(\phi_i, \phi_j) = \sum_{x,y} \left| H_a(\phi_i(x, y)) - H_a(\phi_j(x, y)) \right| \tag{24}$$

where H_a is a Heaviside function defined in (19). This distance is symmetrical and it is irrelevant to the contour size due to shape registration.

2) Sample Clustering and Shape Mode Construction: We construct a graph whose vertices are the aligned shape samples and whose edges are the distances between the aligned samples. The dominant set clustering algorithm [27], which is a novel graph-based clustering algorithm, is used to cluster the samples in order to construct the typical shape modes. The clustering is an iterative bipartition procedure, where a dominant set corresponding to a cluster is split out from the current graph in each iteration step. The number of clusters is automatically determined. Each cluster is a set of similar shapes, corresponding to a typical shape mode.

For each shape mode, a Gaussian is constructed for modeling the level set values at each pixel location [20] using the shape samples corresponding to this mode. This shape model accounts for local variations in shape. The Gaussian probability density function of $\phi(x, y)$ at pixel location (x, y) is:

$$p(\phi(x, y)) = \frac{1}{\sqrt{2\pi}\sigma(x, y)} \exp\left(-\frac{(\phi(x, y) - \phi_M(x, y))^2}{2\sigma^2(x, y)}\right) \tag{25}$$

where $\phi_M(x, y)$ and $\sigma(x, y)$ are, respectively, the mean and the variance of the shape deformations at location (x, y).

3) Shape Mode Transition Matrix: The temporal correlations of shape changes in a periodic motion are modeled using a shape mode transition matrix T which is a $K \times K$ matrix, where K is the number of shape modes. The matrix is estimated based on the clustering results. Each element $T_{i,j}$ in the matrix where i and j index the row and the column of the matrix records the number of the pairs of the consecutive frames where the first frame is assigned to cluster i and the second frame is assigned to cluster j. The elements in each row of T are normalized such that they are summed to unity.

It is noted that matrix T contains the global understanding of the shape changes during a periodic motion, and it permits a more accurate prediction of shapes than the existing autoregressive model-based algorithm [19] which uses local information about shape changes to predict shapes.

B. Shape Mode Prediction

We use the color-based evolution result ϕ_c in the current frame, the shape mode in the previous frame, and matrix T to predict the shape mode in the current frame. This prediction is obtained by maximizing a posterior probability. Let $P(S_t|O_t, S_{1:t-1})$ be the posterior probability of the shape mode S_t in the current frame, given the observations O_t in the current frame, where O_t is represented by ϕ_c in this paper and the shape modes $S_{1:t-1}$ in the previous frames. Then, the optimal shape mode S_t^* in the current frame is determined by:

$$S_t^* = \arg\max_{S_t}(P(S_t|O_t, S_{1:t-1})). \tag{26}$$

In accordance with Bayes' rule, $P(S_t|O_t, S_{1:t-1})$ can be represented as:

$$P(S_t|O_t, S_{1:t-1}) \propto P(O_t|S_t, S_{1:t-1})P(S_t|S_{1:t-1}). \tag{27}$$

We choose to consider Markov chains of order 1 for $P(S_t|S_{1:t-1})$, i.e., it is assumed that $P(S_t|S_{1:t-1})$ is equal to $P(S_t|S_{t-1})$. Then, (27) is transformed to:

$$P(S_t|O_t, S_{1:t-1}) \propto P(O_t|S_t)P(S_t|S_{t-1}). \tag{28}$$

The term $P(S_t|S_{t-1})$ reflects the temporal transition relations between shape modes and it is obtained from matrix T. The term $P(O_t|S_t)$ is the likelihood function which measures the comparability between the predicted shape mode and the current observation. Let $\phi_M^{S_t}$ be the mean shape of shape mode S_t. As the less the distance $d(\phi_c, \phi_M^{S_t})$, the more probable it is that S_t matches the current observation, it can be assumed that the square of the distance $d(\phi_c, \phi_M^{S_t})$ approximately obeys the exponential distribution with a parameter λ. Then, the likelihood function $P(O_t|S_t)$ is estimated as:

$$P(O_t|S_t) \propto \exp\left(-\lambda d(\phi_c, \phi_M^{S_t})^2\right). \tag{29}$$

We calculate the square d_i^2 of the distance from each shape sample i in the cluster of S_t to the mean shape $\phi_M^{S_t}$. According to the maximum likelihood evaluation, parameter λ is estimated as:

$$\lambda = \frac{n}{\sum_{i=1}^{n} d_i^2} \tag{30}$$

where n is the number of samples in the cluster of S_t. Using (26), the optimal shape mode S_t^* in the current frame is selected from all the shape modes.

C. Contour Evolution With the Shape Constraint

The Gaussian models shown in (25) for the mode S_t^* are used to constrain the evolution of the contour. According to [20], [28], the level set speed function under the shape constraint is formulated as

$$F_{\text{shape}}(x, y) = \frac{(\phi(x, y) - \phi_M(A(x, y)))^2}{\sigma(A(x, y))^2} + \log\sigma(A(x, y)) \tag{31}$$

where A is the affine parameters for shape registration, and ϕ_M and σ are the Gaussian parameters in the shape model. The contour is evolved to the boundary with the overall speed $(F_{\text{curv}} + F_{\text{shape}})$ in the normal direction:

$$\frac{\phi^{n+1}(x, y) - \phi^n(x, y)}{\Delta t} = (F_{\text{curv}} + F_{\text{shape}}(x, y))\,|\nabla\phi(x, y)| \tag{32}$$

where the definitions of F_{curv}, n, and Δt are the same as in (2). The initial value of ϕ for the iteration is set to ϕ_c, as ϕ_c contains local information about the true contour in the image. The iterations of (32) continue until a predefined number of iterations is reached or the level set value at each pixel location (x, y) lies within $[\phi_M(x, y) - 2\sigma(x, y), \phi_M(x, y) + 2\sigma(x, y)]$.

VII. Abrupt Motion

We use the method in [39] to detect abrupt motion between the current frame and the previous frame. If abrupt motion is detected, the affine motion parameters are estimated using the particle swarm optimization (PSO) [29], and applied to the contour obtained in the previous frame to obtain the initial contour in the current frame; otherwise the evolution result in the previous frame is used as the initial contour in the current frame. The initial contour is then further evolved to provide the segmentation of the object.

The global motion is represented by a six dimensional affine vector $m_t = (x_t, y_t, \theta_t, s_t, \alpha_t, \beta_t)$ [37] where $x_t, y_t, \theta_t, s_t, \alpha_t, \beta_t$ denote x, y translation, rotation angle, scale, aspect ratio, and skew direction at frame t. The particles $\{m_t^i\}_{i=1}^N$ sample the affine parameters m_t at frame t, where i indexes a particle and N is the number of particles. The particles $\{m_t^i\}_{i=1}^N$ at time t are initialized by sampling from a Gaussian distribution $\eta(m_{t-1}, \Sigma)$, where the mean m_{t-1} is the estimated global motion at time $t-1$, and the covariance Σ is usually set manually. The fitness value of each particle is calculated using $f(m_i^i) = P(O_t^i|m_t^i)$ where O_t^i is the image region specified by particle m_t^i. The probability $P(O_t^i|m_t^i)$ is estimated using the observation model in [24]. Values of pixels in an object image region in a frame are flattened to a vector. An incremental SVD is applied to the vectors for the frames in which the object has been tracked, to incrementally learn the low-dimensional subspace which represents the appearance model of the object. The reconstruction error of O_t^i to the subspace is used to estimate $P(O_t^i|m_t^i)$. Please refer to [24] for details.

Traditional particle filtering algorithms use re-sampling to ensure the quality of the particle set, so that there is a higher probability of finding the object motion [33]. However, a simple re-sampled particle set cannot cover abrupt and arbitrary motions. Thus, we evolve the particles with an adaptive velocity in PSO to deal with abrupt and arbitrary motions. For each particle i, the individual best state d_t^i which has the

maximum fitness value is updated in the n-th PSO iteration using the following Eq.

$$d_t^i = \begin{cases} m_t^{i,n}, & \text{if } f(m_t^{i,n}) > f(d_t^i) \\ d_t^i, & \text{else.} \end{cases} \tag{33}$$

The global best state g_t among all the particles is defined as:

$$g_t = \arg\max_{d_t^i} f(d_t^i). \tag{34}$$

Then, the process for evolving the particles with an adaptive velocity v_t is expressed using the following equations:

$$\begin{cases} v_t^{i,n+1} = w v_t^{i,n} + c_1 u_1 (d_t^i - m_t^{i,n}) + c_2 u_2 (g_t - m_t^{i,n}) \\ m_t^{i,n+1} = m_t^{i,n} + v_t^{i,n+1} \end{cases} \tag{35}$$

where the weight w is adopted to control the influence of the previous velocity $v_t^{i,n}$, c_1 and c_2 are positive constants, and u_1 and u_2 are random values uniformly distributed in [0, 1].

The iterations defined by Eqs. (33)–(35) continue until convergence. The output is the global best state g_t which is the global motion we need to obtain. In this way, the particles cooperate with each other according to their observations to capture the abrupt motion.

We use the contour C_{t-1} at frame t-1 and the estimated global motion $g_t = (x_t, y_t, \theta_t, s_t, \alpha_t, \beta_t)$ at frame t to estimate the initial contour $\tilde{C}_t$ at frame t. Each point $(x, y) \in C_{t-1}$ is transformed into point $(x', y') \in \tilde{C}_t$ using the following Eq.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_t \cos\theta_t & s_t \alpha_t \beta_t \cos\theta_t - \alpha_t \beta_t \sin\theta_t \\ s_t \sin\theta_t & s_t \alpha_t \beta_t \sin\theta_t + \alpha_t \beta_t \cos\theta_t \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \end{bmatrix}. \tag{36}$$

The level set embedding function values $\tilde{\phi}_t$ corresponding to $\tilde{C}_t$ are computed using (1). Then, the level set evolution based on $\tilde{\phi}_t$ is conducted using the color-based or shape-based contour evolution to obtain the final contour.

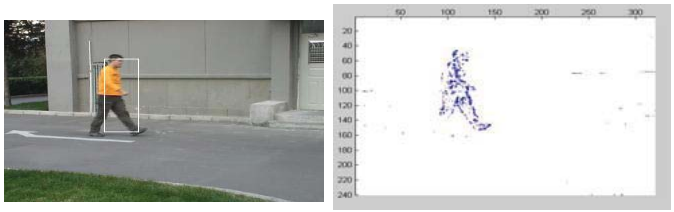
VIII. Experimental Results

All the above algorithms in our framework are implemented using MATLAB on the Windows XP platform. In the following, the performances of our tracking initialization algorithm, our color-based contour evolution algorithm, our adaptive shape-based contour evolution algorithm, our dynamic shape-based contour evolution algorithm, and our algorithm for handling abrupt motions, are evaluated in succession.

A. Contour-Based Tracking Initialization

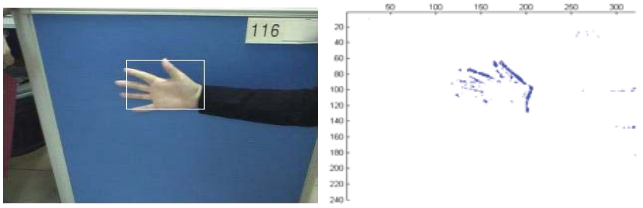
In all the experiments, the contour-based tracking is successfully initialized based on the optical flow detection result in the first frame. Some of the results of the initialization are selected and shown below.

Fig. 1 shows the automatic initialization of the tracking of a walking person, in which (b) is the result of optical flow detection applied to the image (a). The rectangle shown in (a) is the detected motion region with center coordinates (100, 118) and height and width 96 and 34 pixels respectively. The detected motion region is acceptable and the boundary of the rectangle is a good enough initial contour for the color-based contour evolution.



(a) (b)

Fig. 1. Initialization of the tracking of a walking person. (a) Image and detected motion region. (b) Detected optical flow.



(a) (b)

Fig. 2. Initialization of the tracking of an open hand. (a) Image and detected motion region. (b) Detected optical flow.

Fig. 2 shows the automatic initialization of the tracking of a moving open hand, in which (b) is the result of optical flow detection applied to the image (a). The rectangle shown in (a) is the detected motion region with center coordinates (95, 165) and height and width 79 and 60 pixels respectively. The edge of the rectangle is a good enough initial contour for the follow-up color-based contour evolution.

B. Color-Based Evolution

To verify our color-based contour evolution algorithm, we have performed a number of experiments on various sequences captured using moving cameras. In the following, we only show some examples of the tracking results, and then illustrate some comparison results.

In our experiments, we incorporate the fast narrow band approach [2], which constructs a narrow band around each contour and only updates the level set functions of the pixels within the band, to accelerate the evolution of level sets. For all the sequences, the parameter σ in (16) and (17) is set to 0.21 and l in (20) is set to 2.

1) Results on Real Videos: In the first example, a Mickey Mouse head is tracked from frame 1 to frame 155. The challenging point in the video is that the strips, which have the same color as the Mickey Mouse head, strongly perturb the evolution of the contour sections adjacent to the strips. Fig. 3 shows the tracking results. It is seen that the contour of the head is successively tracked: the disturbance from the strips in the background does not influence the evolution of the contour of the Mouse head.

The second is an example of non-rigid object contour tracking. A walking person in an outdoor scene is tracked from frame 1 to frame 74. The colors of some areas in the background are similar to those of the person: for instance, in frame 74, a black region of the background is adjacent to the black hair of the person. The tracking results are shown

Fig. 3. Tracking results of our color-based algorithm for the Mickey Mouse sequence. The frame numbers are 28, 96, 118, and 153, respectively.

Fig. 4. Tracking results of our color-based algorithm for the outdoor person walking sequence. The frame numbers are 24, 55, 68, and 74, respectively.

Fig. 5. Tracking results of our color-based algorithm for the indoor human walking sequence. The frame numbers are 16, 36, 70, and 114, respectively.

in Fig. 4. The contour of the person is successfully tracked. Our algorithm is not sensitive to the background disturbance. In particular, the head, the arms and legs of the person are accurately tracked.

In the third example, a person in an indoor scene is tracked from frame 1 to frame 114. The background is cluttered with some regions similar in color to parts of the person. Fig. 5 shows the tracking results. The contour of the walking person including the arms and legs is still tracked accurately in spite of the background clutter.

2) Comparisons: To show the advantage of employing the MRF in our color-based contour evolution, we compare our algorithm with Yilmaz *et al.*'s algorithm [16], [28], which does not consider correlations between pixel values. We add a smoothing regularization term which depends on the length of the contour into Yilmaz *et al.*'s algorithm to act as a prior probability $P(\Re(I))$: $P(R(I)) = \exp(-L(C)/\lambda)$ where $L(C)$ is the contour length [30] and λ is a parameter whose change corresponds to the change in the weight of the length prior.

Figs. 6 and 7 show the tracking results of Yilmaz *et al.*'s algorithm and the algorithm with the length prior for the Mickey Mouse head sequence, where λ is set to 400. From the comparison between Figs. 3, 6, and 7, it is seen that the contours obtained using Yilmaz *et al.*'s algorithm and the algorithm with the length prior are less accurate: the contours obtained using our color-based algorithm are much smoother than the contours obtained using the two competing algorithms. The disturbance from the strips in the background obviously influences the tracking results of the two competing algorithms.

We contrast the tracking results of our algorithm, Yilmaz *et al.*'s algorithm, and the algorithm with the length prior with the ground truth which we have labeled by hand. We use the common area rate AR to evaluate the accuracy of the obtained contour in each frame: $AR = (S_{\text{com}} - S_1 - S_2)/S_{\text{truth}}$,

Fig. 6. Tracking results of Yilmaz *et al.*'s [28] algorithm for the Mickey Mouse head sequence.

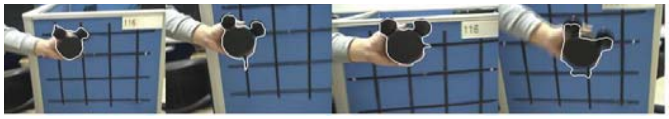


Fig. 7. Tracking results of the algorithm with the length prior for the Mickey Mouse head sequence.

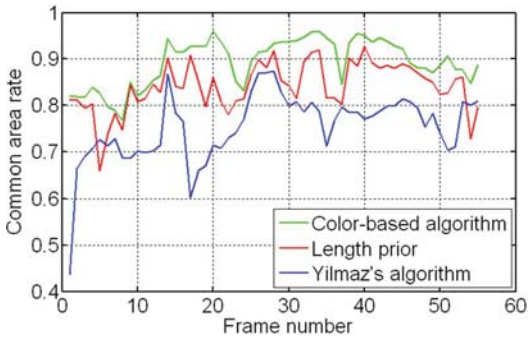


Fig. 8. Common area rates of our color-based algorithm, Yilmaz *et al.*'s [28] algorithm, and the algorithm with the length prior for the Mickey Mouse head sequence.

where S_{truth} is the pixel area within the ground truth contour, S_{com} is the common area within both the ground truth contour and the evaluated contour, S_1 is the area within the ground truth contour and outside the evaluated contour, and S_2 is the area within the evaluated contour and outside the ground truth contour. The higher the common area rate, the more accurate the evaluated contour. Fig. 8 shows the common area rate curves of our algorithm, Yilmaz's algorithm, and the algorithm with the length prior for the Mickey Mouse head sequence, where each curve is obtained by connecting the points representing the common area rates for each frame in the sequence order. It is obvious that our algorithm has a higher common area rate than both the competing algorithms in each frame, while the algorithm with the length prior has a higher common area rate than Yilmaz's algorithm.

Fig. 9 shows the results of Yilmaz *et al.*'s algorithm for the outdoor person walking sequence. It is seen that Yilmaz's algorithm does not effectively handle the background disturbance such as in frame 74. Fig. 10 compares the common area rate curves of our algorithm and Yilmaz's algorithm for the outdoor person walking sequence. It is shown that the contour obtained using Yilmaz *et al.*'s algorithm is less accurate than the contour obtained using our color-based algorithm.

C. Adaptive Shape-Based Contour Evolution

Our adaptive shape-based contour evolution algorithm is tested on videos which are captured using mobile cameras. The videos contain partial occlusions besides background

Fig. 9. Tracking results of Yilmaz *et al.*'s [28] algorithm for the outdoor person walking sequence.

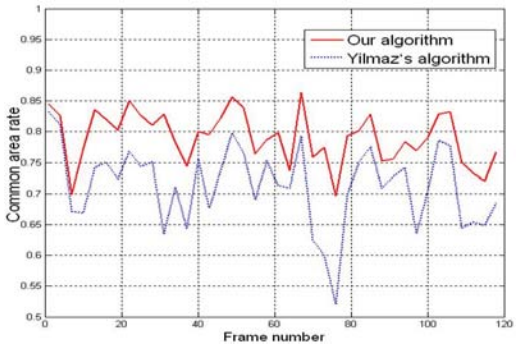


Fig. 10. Common area rates of our algorithm and Yilmaz *et al.*'s [28] algorithm for the outdoor walking person sequence.

Fig. 11. Tracking results of our adaptive shape-based algorithm for the Mickey Mouse head sequence.

Fig. 12. Results of tracking a face occluded by a hand using the color-based evolution alone. The frame numbers are 243, 247, 251, and 254, respectively.

disturbance and motion blurring. The shape samples are registered and then transformed to 3000-dimensional vectors, and the first 20 singular vectors are chosen as the eigenbasis. The threshold for the Mahalanobis distance-based criterion is set to 16. The contour tracking results are used to update the shape model every four frames when the shape model is not introduced into the contour evolution, and once a frame when the shape model is introduced.

In the first example, the Mickey Mouse head sequence is used. The parameter β in (23) is set to 100 for this sequence. As shown in Fig. 11, although there is background disturbance making the contrast between object and background in some contour sections very weak, and around the 151th frame there is severe motion blur, the contour of the head is still tracked accurately and robustly. On comparing Figs. 3 and 11, it is seen that our adaptive shape-based algorithm yields more accurate results than our color-based algorithms. So, introduction of the shape model can improve the tracking accuracy.

In the second example, a moving face is tracked. In this sequence, the camera zooms and moves when the person changes her face's pose continuously: for example, from frame

Fig. 13. Results of tracking the face using our adaptive shape-based evolution. The frame numbers are 59, 65, 107, and 118, respectively.

Fig. 14. Results of tracking the face and the hand separately using our adaptive shape-based evolution.

Fig. 15. Tracking the contour of a face occluded by another face. From left to right, the frame numbers are 387, 432, 440, and 470, respectively.

50 to frame 118, the face pose changes from looking ahead to looking downwards, leading to shape changes. In a number of frames, the face is seriously occluded by a moving hand which has a color similar to that of the face. The parameter β in (23) is set to 200 for this sequence. Fig. 12 shows the results of tracking the face occluded by the hand using the color-based evolution alone. During the occlusions, the face and the hand are tracked as a single region bounded by one contour. Fig. 13 shows the results of tracking the face using our adaptive shape-based evolution algorithm. It is clear that the face's contour section which is occluded by the hand is recovered successfully using the shape prior, and due to the incremental updating of the shape model, the shape changes of the faces are effectively learned by the shape model. Fig. 14 shows the results of tracking the face and the hand separately using our adaptive shape-based evolution algorithm. During the time that the face is occluded by the hand, the contours of the face and the contours of the hand are successfully tracked: the contours tightly enclose the face and the hand respectively.

The third example is widely used to test general object tracking algorithms. A girl changes her facial pose over time under varying lighting conditions. In the middle of the video, the girl's face is severely occluded by a man's head. Fig. 15 shows the results of our adaptive shape-based algorithm for tracking the face of the girl. It is seen that our algorithm tracks the contour of the face successfully. Fig. 16 shows the results of tracking the two faces separately. The contours of the two faces are both successfully tracked, during the time that the face of the girl is occluded by the face of the man.

The fourth example in which large shape changes occur is used to stress the learning ability of our adaptive shape model. In this sequence, an open hand moves where in many frames it is occluded by an object. The fingers are initially separated. They are brought together as the sequence progresses. Although the palm stands out by its color cue,

Fig. 16. Results of tracking two faces separately using our adaptive shape-based evolution algorithm. From left to right, the frame numbers are 423, 441, 463, and 469, respectively.

Fig. 17. Tracking results of our adaptive shape-based algorithm for the open hand sequence. The frame numbers are 38, 45, 110, and 117, respectively.

Fig. 18. Tracking results of our color-based algorithm for the open hand sequence.

very large shape changes make this sequence very suitable for testing the robustness of the shape models. The parameter β in (23) is set to 150 for this sequence. First, the shape samples for the palm with separated fingers are used to construct the shape model. Second, the trained shape model is used to conduct contour evolution and recover the occluded part of the contour, and the shape model is updated online. Third, when the shape of the palm gradually changes to the new one with fingers brought together, the new shape is learned online by the shape model and the updated model is used to recover the occluded part of the contour of the palm with the new shape. Fig. 17 shows the tracking results. The contour is correctly tracked in all the frames. It is seen that even though the new shape with fingers brought together appears as shown in frames 110 and 117, a good track of the contour of the occluded palm is kept, showing the good adaptability of our shape model.

Fig. 18 shows the tracking results of our color-based contour evolution algorithm for this moving palm sequence. It is seen that only the un-occluded part of the contour is tracked: the occluded part of the contour is not recovered. The color feature alone is not sufficient to perform tracking during partial occlusions, and the shape priors are needed. Fig. 19 shows the tracking results with the shape prior but without shape subspace updating during the tracking. At first, as the shape of the object does not change very much, the tracking results are satisfactory. However, when the fingers are brought together from frame 100 onwards, the shape prior cannot provide the correct information, and the contour of the fingers is not correctly tracked. From this comparison, it is seen that our algorithm with an adaptive shape subspace model keeps good track of objects which undergo large changes in shape.

D. Dynamic Shape-Based Contour Evolution

For testing our dynamic shape-based contour evolution, contours of walking persons are tracked under noise, partial occlusions, and background clutter, etc. In the following, we demonstrate dynamic shape-based tracking results on real videos, and then illustrate some comparison results.

Fig. 19. Tracking results of the algorithm without shape subspace updating for the open hand sequence.

Fig. 20. Tracking results for the sequence disturbed by salt and pepper noise. The frame numbers are 17, 23, 28, and 32, respectively.

Fig. 21. Tracking results for the sequence with occlusions by ellipses. The frame numbers are 17, 23, 28, and 32, respectively.

Fig. 22. Tracking results of our dynamic shape-based algorithm for the indoor walking person sequence. The frame numbers are 69, 76, 93, and 96, respectively.

1) Results on Real Video Sequences: The first sequence is captured from an outdoor scene. With respect to this sequence, shape samples are trained to obtain 6 typical and representative shape modes. Figs. 20 and 21 show the tracking results of our dynamic shape-based contour evolution algorithm for the sequences with two computer generated perturbations: the first is that the frames are disturbed by salt-pepper noise and the second is that the person is partially occluded in each frame by an ellipse which is placed randomly around the tracked person. It is seen that for both the noise perturbation shown in Fig. 20 and the partial occlusion perturbation shown in Fig. 21, our algorithm keeps good track of the contour of the walking person. Our dynamical shape model is thus effective for recovering missing or perturbed sections of the contour of the tracked object.

The second sequence used corresponds to Fig. 5. Nine shape modes are discovered using the dominant set clustering and the shape mode transition matrix is calculated. Fig. 22 shows the tracking results of our dynamic shape-based algorithm for this sequence. It is shown that the walking person is successfully tracked in all the frames.

2) Comparison With the Autoregressive Model: We compare our dynamic shape-based algorithm with the autoregressive model-based algorithm [19] which is the only existing algorithm for contour tracking using a dynamic shape model.

Fig. 23. Tracking results of the autoregressive model-based algorithm for the sequence with salt and pepper noise.

Fig. 24. Tracking results of the autoregressive model-based algorithm for the indoor walking person sequence. The frame numbers are 69, 76, 93, and 96, respectively.

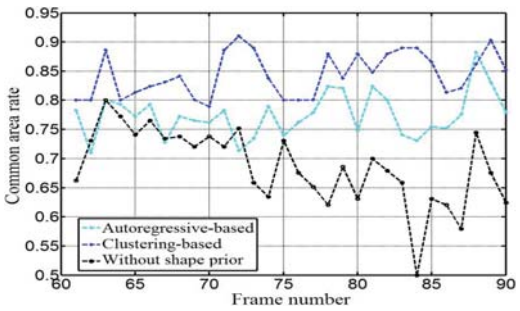


Fig. 25. Common area rates for the indoor walking person sequence.

Fig. 23 shows the results of the autoregressive model-based algorithm for tracking the outdoor walking person in the sequence with salt and pepper noise, where the autoregressive model is updated in batch mode using the final contour evolved in each frame. From the comparison between Figs. 20 and 23, it is seen that the results obtained using the autoregressive model-based algorithm are less accurate than the results obtained using our dynamic shape-based algorithm.

Fig. 24 shows the tracking results of the autoregressive model-based algorithm for tracking the walking person in the indoor scene. From the comparison between Figs. 22 and 24, it is seen that the results of the autoregressive model-based algorithm are much less accurate than the results of our algorithm. We have manually constructed the ground truth contours for this sequence. Fig. 25 shows the common area rates of our algorithm, the autoregressive model-based algorithm, and our color-based algorithm for this sequence. It is seen that the area rate of our dynamic shape-based algorithm is much higher that those of the autoregressive model-based algorithm and the color-based algorithm. It clearly illustrates that our dynamical shape-based algorithm effectively models the shape changes in periodic motions.

The reasons why our algorithm obtains more accurate results than the autoregressive model-based algorithm are as follows. The autoregressive model-based algorithm uses an autoregressive model to approximate the temporal relations between the shape samples. The object shape in the current frame is dependent on the shapes in a few previous frames and it is predicted using the shapes in these previous frames. This algorithm models the samples only using data fitting, supplies the shape prior using the local temporal information, and depends heavily on the periodic property of data. When the shapes obtained in the previous frames have some deviations in contrast to the closest matched shape samples, the shape may be inaccurately predicted (See frames 76 and 93 in Fig. 24). However, our algorithm clusters shape samples into several shape modes. The temporal correlations of shape changes are modeled using a shape mode transition matrix which contains a high level of understanding of the shape samples and their temporal changes. Our algorithm predicts the object shape using the global temporal information and then obtains a more accurate shape prior.

Fig. 26. Results of our PSO-based algorithm for tracking a face undergoing rapid motion. The frame numbers are 3, 13, 19, and 25, respectively.

Fig. 27. Results of the algorithm without abrupt motion handling for tracking a face undergoing rapid motion.

Fig. 28. Results of the particle filter-based algorithm [33] for tracking a face undergoing rapid motion.

E. Contour Evolution With Abrupt Motion

For validating the effectiveness of our PSO-based algorithm for handling abrupt motion, a number of tracking experiments as well as comparison results are carried out on both abrupt motion and low frame rate videos. The covariance matrix of the Gaussian used to initialize particles is set to $\Sigma = diag(8^2, 8^2, 0.02^2, 0.01^2, 0.002^2, 0.001^2)$, and the number of particles is set to 200.

The first video sequence contains a human face with a rapid motion. Fig. 26 shows the results of our PSO-based algorithm for tracking the face. The contour obtained using our algorithm fits tightly to the face throughout the whole tracking process. The successful tracking rate is 100%. To show the advantage of our PSO-based algorithm, we compare the results of our PSO-based algorithm with the results without abrupt motion handling and the results of the particle filtering-based algorithm. Fig. 27 shows the tracking results without abrupt motion handling. The algorithm quickly loses track of the object in frame 3 and never recovers the track. The contour of the face is tracked successfully for only two of the 26 frames, giving a success rate of 7.6%. So, special handling of abrupt motion is necessary for videos with abrupt motions. We replace the PSO algorithm in our algorithm with the particle filtering algorithm [33] to construct the initial contour which is further evolved using our color-based algorithm to produce the final contour. Fig. 28 shows the tracking results of the particle

Fig. 29. Tracking results of our PSO-based algorithm for the low-frame-rate video. The frame numbers are 4, 13, 21, and 27, respectively.

Fig. 30. Tracking results of our color-based algorithm without abrupt motion handling for the low-frame-rate video.

Fig. 31. Tracking results of the particle filter-based algorithm for the low-frame-rate video.

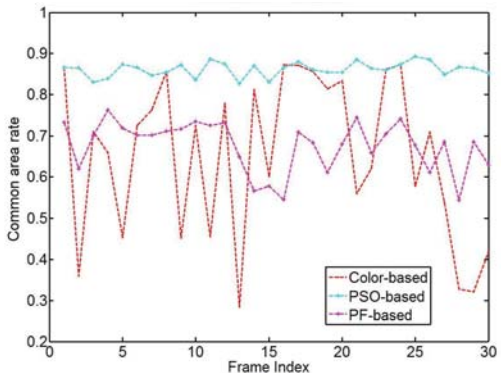


Fig. 32. Common area rates of the algorithms with or without abrupt motion handling.

filter-based algorithm [33]. It is apparent that the contour of the face is tracked successfully for only 7 frames; the contour of the face is mistakenly tracked for 13 frames; and the face is lost for tracking for 6 frames. The successful tracking rate of the particle filter-based algorithm is 26.9%. This means that the particle filtering algorithm cannot deal effectively with abrupt motions. This is because it does not have the PSO algorithm's adaptive velocity which is the essential element for dealing with abrupt motions.

We use the low frame rate to simulate abrupt motion for the Mickey Mouse head sequence. One frame is sampled from every 10 frames in the original sequence to form a new sequence with a low frame rate. Fig. 29 shows the tracking results of our PSO-based algorithm for this sequence. It is seen that the information discontinuity between consecutive frames in this video does not affect our proposed algorithm: a smooth contour is obtained in each frame. Fig. 30 shows the tracking results of our color-based contour evolution algorithm without abrupt motion handling. This algorithm cannot adjust to the information discontinuity between consecutive frames, leading to failures in the tracking. The reason for this is that the low frame rate causes that the evolved contour in the previous frame contains a small part of the true contour in the current frame. As a result, the evolved contour runs into a local minimum. Fig. 31 shows the tracking results of the particle filter-based algorithm [33] for this video. The obtained contours are unsatisfactory. Fig. 32 shows the common area rates of the algorithms with and without abrupt motion handling for this sequence. It is seen that the common area rates of our PSO-based algorithm are much higher than those of the algorithms without abrupt motion handling.

IX. Conclusion

In this paper, we have presented an effective framework for tracking object contours. We have the following conclusions: 1) Our color-based contour evolution algorithm which applies the MRF theory to model the correlations between pixel values for posterior probability estimation is more robust to background disturbance than the region-based method which does not consider correlations between the values of neighboring pixels for posterior probability estimation. 2) Our adaptive shape-based contour evolution algorithm, which efficiently fuses the global shape information and the local color information and uses a flexible shape model updating algorithm, is robust to partial occlusions, weak contrast at the boundaries, and motion blurring, etc. 3) Our dynamical shape prior model effectively characterizes the temporal correlations between contour shapes in periodic motions, and thus it obtains more accurate contours than the existing autoregressive model. 4) Our PSO-based algorithm can deal effectively with contour tracking for videos with abrupt motions, and it outperforms the particle filter-based algorithm.

References

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[2] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *J. Comput. Phys.*, vol. 118, no. 2, pp. 269–277, 1995.

[3] S. Osher and J. A. Sethian, "Fronts propagation with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, 1988.

[4] B. Horn and B. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, Aug. 1981.

[5] X. Zhou, W. Hu, Y. Chen, and W. Hu, "Markov random field modeled level sets method for object tracking with moving cameras," in *Proc. Asian Conf. Comput. Vis.*, 2007, pp. 832–842.

[6] M. Partio, B. Cramariuc, M. Gabbouj, and A. Visa, "Rock texture retrieval using gray level co-occurrence matrix," in *Proc. Nordic Signal Process. Symp.*, Oct. 2002, pp. 1–5.

[7] C. Stauffer and W. E. L. Grimson "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Jun. 1999, pp. 246–252.

[8] J. A. Sethian, "Level set methods and fast marching methods: Evolving interfaces in computational geometry," in *Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[9] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[10] D. X. Xu, J. N. Hwang, and C. Yuan, "Segmentation of multi-channel image with Markov random field based active contour model," *J. VLSI Signal Process.*, vol. 31, no. 1, pp. 45–55, May 2002.

[11] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Join Conf. Artif. Intell.*, 1981, pp. 674–679.

[12] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *Int. J. Comput. Vis.*, vol. 22, no. 1, pp. 61–79, Feb. 1997.
[13] N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 3, pp. 266–280, Mar. 2000.
[14] T. Bailloeul, "Active contours and prior knowledge for change analysis: Application to digital urban building map updating from optical high resolution remote sensing images," Ph.D. dissertation, National Laboratory of Pattern Recognition, Inst. Automation, Chinese Academy of Sciences, Bejing, China, Oct. 2005.
[15] S. C. Zhu and A. Yuille, "Region competition: Unifying snakes, region growing and bayes/MDL for multiband image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 9, pp. 884–900, Sep. 1996.
[16] A. Yilmaz, X. Li, and M. Shah, "Object contour tracking using level sets," in *Proc. Asian Conf. Comput. Vis.*, 2004, pp. 1–7.
[17] Y. Shi and W. C. Karl, "Real-time tracking using level sets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2. May 2005, pp. 34–41.
[18] M. Leventon, E. Grimson, and O. Faugeras, "Statistical shape influence in geodesic active contours," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun. 2000, pp. 316–323.
[19] D. Cremers, "Dynamical statistical shape priors for level set based tracking," *IEEE Trans. Patten Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1262–1273, Aug. 2006.
[20] N. Paragios and M. Rousson, "Shape priors for level set representations," in *Proc. Eur. Conf. Comput. Vis.*, 2002, pp. 78–92.
[21] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Comput. Vis. Image Understand.*, vol. 61, no. 1, pp. 38–59, Jan. 1995.
[22] M. Fussenegger, P. M. Roth, H. Bischof, and A. Pinz, "Online, incremental learning of a robust active shape model," in *Proc. DAGM-Symp. Pattern Recognit.*, 2006, pp. 122–131.
[23] D. Skocaj and A. Leonardis, "Weighted and robust incremental method for subspace learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Oct. 2003, pp. 1494–1501.
[24] J. Lim, D. Ross, R. S. Lin, and M. H. Yang, "Incremental learning for visual tracking," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004, pp. 793–800.
[25] Y. Li, "On incremental and robust subspace learning," *Pattern Recognit.*, vol. 37, no. 7, pp. 1509–1518, 2004.
[26] N. Paragios, M. Rousson, and V. Ramesh, "Matching distance functions: A shape-to-area variational approach for global-to-local registration," in *Proc. Eur. Conf. Comput. Vis.*, 2002, pp. 775–789.
[27] M. Pavan and M. Pelillo, "A new graph-theoretic approach to clustering and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun. 2003, pp. 18–20.
[28] A. Yilmaz, X. Li, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1531–1536, Nov. 2004.
[29] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Nov.–Dec. 1995, pp. 1942–1948.
[30] Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984.
[31] D. Cremers, N. Sochen, and C. Schnorr, "Towards recognition-based variational segmentation using shape priors and dynamic labeling," in *Proc. 4th Int. Conf. Scale Space Methods Comput. Vis.*, 2003, pp. 388–400.
[32] A. R. Mansouri, "Region tracking via level set PDEs without motion computation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 947–961, Jul. 2002.
[33] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi, "Tracking deforming objects using particle filtering for geometric active contours," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1470–1475, Aug. 2007.
[34] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[35] J. Kwon and K. Lee, "Tracking of abrupt motion using Wang-Landau Monte Carlo estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 387–400.
[36] D. Cremers, M. Rousson, and R. Deriche, "A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape," *Int. J. Comput. Vis.*, vol. 72, no. 2, pp. 195–215, Apr. 2007.
[37] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, no. 1, pp. 125–141, May 2008.
[38] T. Riklin-Raviv, N. Kiryati, and N. Sochen, "Segmentation by level sets and symmetry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1. Jun. 2006, pp. 1015–1022.
[39] J. Sullivan and J. Rittscher, "Guiding random particles by deterministic search," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 1. Jul. 2001, pp. 323–330.
[40] B. Jung and G. S. Sukhatme, "Real-time motion tracking from a mobile robot," *Int. J. Soc. Robot.*, vol. 2, no. 1, pp. 63–78, Mar. 2010.
[41] C. Sagiv, N. A. Sochen, and Y. Y. Zeevi, "Integrated active contours for texture segmentation," *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1633–1646, Jun. 2006.
[42] S. V. Pons, J. L. G. Rodriguez, and O. L. V. Perez, "Active contour algorithm for texture segmentation using a texture feature set," in *Proc. Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
[43] C. Bibby and I. Reid, "Robust real-time visual tracking using pixel-wise posteriors," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 831–844.

Weiming Hu, photograph and biography are not available at the time of publication.

Xue Zhou, photograph and biography are not available at the time of publication.

Wei Li, photograph and biography are not available at the time of publication.

Wenhan Luo, photograph and biography are not available at the time of publication.

Xiaoqin Zhang, photograph and biography are not available at the time of publication.

Stephen Maybank, photograph and biography are not available at the time of publication.