ACEEE Int. J. on Communications, Vol. 4, No. 1, July 2013

Implementation of Object Tracking for Real Time Video

Rajesh Chhikara[1], Sunil Kumar[2], Gopinath S[3]
[1]Research Scholar, Singhania University, Rajasthan, India
Chhikara77@rediffmail.com
[2]Maharaja Agrasen Institute of Technology, Delhi, India
Email: sunil.mathur10@gmail.com
[3] Robert Bosch Engineering and Business Solutions Ltd, Banglore, India
Email: selgopi@gmail.com

***Abstract*—Real-time tracking of object boundaries is an important task in many vision applications. Here we propose an approach to implement the level set method. This approach does not need to solve any partial differential equations (PDFs), thus reducing the computation dramatically compared with optimized narrow band techniques proposed before. With our approach, real-time level-set based video tracking can be achieved.**

***Index Terms*—Object Tracking, histogram, blob pruning, adaptive histogram, and two cycle algorithm.**

I. Introduction

Real-time tracking of object boundaries is an important task in many vision applications such as video surveillance, video conferencing, and human-computer interaction, etc. Parametric contours have been applied successfully to achieve real-time performance [12, 7, 6, 19], but they have difficulties in handling topological changes such as the merging and splitting of object regions. For this purpose, the level set method [15] is a more powerful technique and various models have been proposed [2, 16, 4, 14, 9, 21]. But the high computational cost of the level set method has limited its popularity in real-time scenarios. In this paper, we propose a novel approach to implement the level set method. Our approach does not need to solve any partial differential equations (PDEs), thus reducing the computation dramatically compared with optimized narrow band techniques proposed before. With our approach, real-time level-set-based video tracking can be achieved.

Tracking models using level sets can be classified as edge-based or region-based. In [16], a geodesic model that combines motion and edge information was proposed. Such edge-based models can be traced back to the snake model in [13]. Based on morphing images, an early work on region-based tracking was proposed in [2]. Using the difference between the current frame and a reference background, a region-based model was proposed in [4]. By generalizing the region competition[22] idea, statistical approaches become quite popular recently. In [14, 21], the feature distributions of both the object and background regions were used for tracking. In [9], a predefined distribution for the object region was tracked by minimizing a Kullback-Leibler or Bhattacharyya distance.

Considerable work has been done to improve the speed of level-set-based curve evolution. The basic idea has been to limit the solution of the level set PDE to a narrow band around the zero level set[1, 17, 20, 16], but issues such as narrow band construction, reinitialization and step size control still make existing level set methods computationally too expensive for real-time tracking.

The novel implementation we propose in this paper avoids the above problems. Our approach is based on the key observation that implicitly represented curves can be moved by simply switching elements between two linked lists. We also propose a Gaussian filtering process to replace curvature-based smoothing, and this further speeds up our algorithm. Another advantage of our method is that ideas for controlling topological changes in [11] can be incorporated into our algorithm easily, which can be important for the tracking of multiple objects and medical imaging applications. Probabilistic color distribution models have been used for real-time face tracking in video telephony [1] and visual control of computer games [2]. Color histogram back projection [3] is a low complexity, active vision algorithm for finding objects in complex scenes. Kahn et al. [4] combined color histogram back projection with edge, motion and pixel disparity for real-time gesture recognition in the so called Perseus system. Color histogram back projection is a tracking method used for a mobile robot vision system known as Gargoyle [5].

We will do describe tools which enhance the suitability of color histogram back projection for multiple object tracking. Instead of the fixed histogram method used in [3] and [6] we introduce an adaptive algorithm, updating the histogram of the tracked object in order to account for changes in color, illumination, pose and scale. Tracked objects appear as "blobs" in the back projection image. We will developed a fast blob detector using row and column intensity projections in the back projection image. Empirically determined thresholds, blob area estimates and simple motion prediction are combined to resolve occlusion ambiguities, including "color occlusion" where objects have similar color. In their Kalman tracking system, Seo et al. [7] used color histogram back projection for occlusion reasoning, but only to distinguish objects of different color.

With our fast level set implementation, we will developed

DOI: 01.IJCOM.4.1.1

a region-based real-time video tracking system requiring only a small fraction of the CPU power on a standard PC, leaving computational power available for other tasks, such as recognition, trajectory analysis, etc.

II. Tracking Using Adaptive Color Histogram Back Projection

Most of the algorithms available for segmenting objects in complex video scenes require significant processing and fail to solve the focus of attention problem in a scene where many objects are to be segmented and tracked at the same time. The focus of attention problem, that is, finding the region and object of interest in a complex scene was addressed in [3] and extended in [6] using color histogram back projection algorithms. In both cases a fixed model of the object of interest was used to search for the object. In a video sequence, our task is to segment and track multiple objects while at the same time paying attention to variations such as color, illumination, pose and scale. We describe here an adaptive back projection algorithm which accounts for these variations, and which also handles problems of occlusion, enabling unique identification of objects in complex video scenes. Figure 1 shows the system diagram for this algorithm.

A. Initial training model position and histogram

The system is interactive so we avoid the usual difficulties of object detection and tracker initialization encountered in automatic systems. Offline and real-time versions were implemented. In the off-line version, the video sequence is held in a buffer and to initialize the tracking, we use a mouse to define "rubber banded" rectangular bounding boxes around the objects to be tracked from the first frame. The dimensions of these regions are kept for future use. Each box defines an object "training model" which is characterized by its color histogram on a quantised color space; 24bit RGB sample (8 bits in each red, green, blue) images and 256 color histogram bins. Once tracking is initialized, further capturing of new training model histograms is done on-line by software based on position and motion cues. The real-time version supports a video camera and a frame grabber linked to the tracking software so that live video can be used for tracking. Here objects being tracked have to be cooperative, i.e. hold still, while the box contents are captured in the first frame.

Accurate segmentation is not required to track a person; typically the box surrounding the person's shirt, and perhaps head, is adequate. More care is required with small objects like a ball, where background pixels in the box could dominate.

B. Color histogram back projection

Color histogram back projection works as follows [3]. Let M be the histogram of the object of interest, or training "model", and I be the histogram of the image frame. Compute the ratio min(Mi/li,1), where i denotes the histogram bin. This is a confidence measure of how much the color i is characteristic of the model. Then a gray-scale back projection image is constructed from the frame by replacing each pixel with color i by its scaled confidence measure, 255 min(Mi/li,1). In the back- projection image, the model appears as a bright "blob". Figure 2 shows an example where the object is the orange shirt worn by one of the authors (JIA) in a frame from an outdoor sequence.

When tracking several (N say) objects we compute the histogram of the frame I as usual, and the histogram of each model Mj for j = 1, . .N. Then we construct N back projection images, one for each model. Bearing in mind that the sizes of the models are much smaller than the frame sizes, the additional overhead is inconsequential.

C. Row and column projections and blob selection

In [3] a convolution mask, typically circular, is used to estimate the position of the object on the back projection image. We have developed more efficient algorithms [14, 15] based on row and column projections, as shown in Figure 2. In this example the location of the object of interest is unambiguous, and to a good approximation the coordinates of the centroid correspond to the peaks in these projections. In general however there may be other blobs due to similarly coloured objects; see Figure 3. Our problem then is to identify the correct blob automatically.

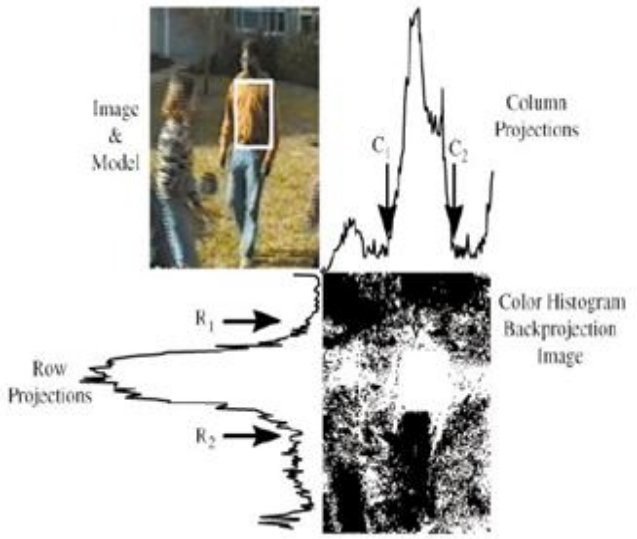


Figure 1. Example of color histogram back projection and row and column projections. The model histogram is extracted from the region inside the white box

We experimented with several statistical methods for initial blob selection. These include the cumulative frequencies of row and column projections and their direct means. We opted for the following technique which is both simple and efficient. First we smooth the row and column projections with a five dimensional averaging filter. This filter eliminates any very narrow peaks in the column and row projections, but retains information from blobs that are significant, i.e. the blob of interest plus any others due to large objects (people) with similar colors.

The smoothed projections are then thresholded. In the current system we use the means of the row and column projections as thresholds; thus the thresholds are adaptive. The row and column projections of a tracked object are

DOI: 01.IJCOM.4.1.1

roughly Gaussian in shape (see Figure 2). We have tested ways to improve the choice of thresholds using estimates of the variances of these distributions, but have found that using the means as thresholds is adequate.

Candidate blobs are then identified as those regions with at least five connected points above the thresholds in both the smoothed row and column projections. An example of a candidate blob selected by this method is shown in Figure 2. It is defined by the rectangular region with corners (R1, C1) , (R1, C2), (R2, C1), and (R2, C2). Here, this method of segmentation produces a larger or smaller than expected rectangular blob (about 2 to 7 pixels in each dimension). This is not a problem as exact edge detection is not the goal.

A major advantage of this method is that the segmentation adapts to the new size of the tracked region, thereby accounting for changes in scale.

D. Blob pruning

At this point there may be one, several or no candidate blobs. In the last frame the tracked object may have left the field of view, or could be occluded by an object of a different color. Whenever object occlusion occurs, we repeat the algorithm using the previous model on the next frame, and re-use the model until the object is once again visible. This effectively implements a criterion to "wait for the object'' to emerge from occlusion. It is efficient as long as occlusion does not persist. In the case of color occlusion, similarly colored objects will appear as candidate blobs, and sometimes these blobs will merge when one or more objects collide or shield each other. This occurs in contact sports such as rugby league where players wear the same colored jerseys.

Distance measure. Let di be the distance between the centroid of the ith candidate blob and the centroid of the blob in the previous back projection, Ai its area, and Ap the blob area in the previous back projection. We then list the blobs in order of increasing (combined) distance:

$$S_t = d_i |A_p - A_j|$$

In most cases the blob with the minimum combined distance corresponds to the tracked object, but to resolve any ambiguities we apply an extra criterion, similar to local block matching. For each candidate blob in the jth frame we check that the direction of motion that this blob has moved relative to the previous frame is consistent with motion prediction based on the velocities;

$$\boldsymbol{v_x = xpos(j-1) - xpos(j-2)}$$
$$\boldsymbol{v_y = ypos(j-1) - ypos(j-2)}$$

estimated from the positions (xpos(j71), ypos(j71)) and (xpos(j72), ypos(j72)) of the centroids of the blobs in the previous two back projections. There are of course special cases where this will not work, e.g. when two objects with the same size and same colors collide and recoil. In this case it is not possible to distinguish between these after collision.

E. Updating model position and histogram

Once the blob corresponding to the tracked object is identified, we prune the interfering candidate blobs by placing a window around the selection, recomputing the row and column projections locally and deriving a new rectangular box which defines the updated model region.

The centroid of the new rectangle is passed to the image display software to update the highlight information on screen. A new model histogram is extracted from the region in the image frame bounded by the new rectangle.

Thus the algorithm adapts not only to the size of the object, as noted above, but also color, illumination and pose changes. The updated model position and histogram are used to find the object in the next frame.

F. Multi-object tracking with a pre-defined model

The process described above is repeated for all N back projection images. It works for tracking objects with different or similar colors. In the latter case one can take advantage of the color similarity to reduce the computational load, especially if one wishes to track all objects of similar color simultaneously. This would be of interest, e.g. in tracking all the players on the same team.

Here we might have a pre-defined model histogram of the team jersey already stored, or have acquired a histogram from one of the players jerseys. For the sake of illustration, we label the three blobs in Figure 3(d), as O0, O1, and O2 respectively. To identify for example the object from blob O0 properly, we use two descriptors, the object position in the previous frame (O00) and the location of the blob in the present frame. This means, it has moved from O00 to O0.

Therefore, O1 and O2 are interferences. Because of their positions, they affect the row and column means from the back projection, leading to a detection for a false position for the target object. Observe however that the distance travelled by the target object between two frames, d0 is much smaller than the distances to the interfering objects d1 and d2 respectively. The algorithm to correct for interfering objects uses the information from Eqns (1) and (2) to isolate the target.

G. Tracking Model

We use a region-based tracking model based on the region competition [22] idea, but our fast level set implementation is also applicable for edge-based models.

We assume each scene of the video sequence is composed of a background region &!0 and M object regions Ω 1, Ω 2, Ω M. The boundaries of these M object regions are denoted as C1, C2 and CM. We model each region with a feature distribution p(v/ Ωm) m= 0; 1; _ _ _ ;M), where v is the feature vector defined at each pixel. For example, the features can be the color, the output of a filter bank designed to model textures, or other visual cues. Assuming that the feature distribution at each pixel is independent, the tracking result of each frame is the minimum of the following region competition energy:

$$E = \frac{\sum_{m=0}^{M} \int_{\Omega_m} \log p(v(x))|\Omega_m)\, dx}{d\lambda \sum_{m=1}^{M} \int_{C_m} ds} +$$

DOI: 01.IJCOM.4.1.1

λ is the non-negative regularization parameter.
By computing the first variation of this energy, the curve evolution equation for the minimization of this energy is:

$$\frac{dC_m}{dt} = (F_d + F_s)\vec{N}C_m \quad (m = 1,2, \ldots . M)$$

Where $\vec{N}C_m$ is the normal of C_m pointing outward, and Fd and Fs are the speed resulting from Fd and Fs, respectively. The speed Fd represents the competition between two regions an it is

$$F_d = \log[p(v(x)|\Omega_m / p(v(x)|\Omega_{out})]$$

where Ω_{out} out denotes the region outside Cm at $x \in C_m$. The speed Fs makes the curve smooth and it is Fs = $\lambda\kappa$, where κ is the curvature.

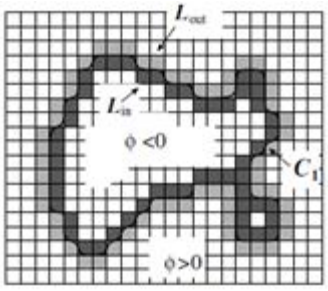


Figure 2. The implicit representation of the curve C1 and the two lists Lin and Lout in the neighborhood of C1

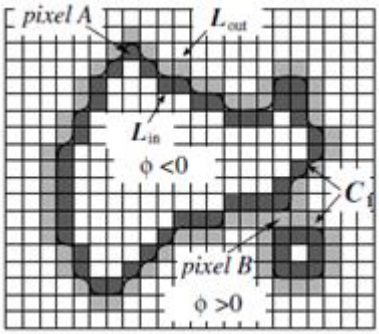


Figure 3. Illustration of the motion of the curve C1 by switching pixels between Lin and Lout

Since the focus of this paper is on real-time level set implementation for video tracking, we use a simple tracking strategy as follows. For each frame, we use the tracking results from the last frame as the initial curves, and then evolve each curve according to (2) to locate the object boundaries in the current frame. Once it stops, we move on to track the next frame of the video sequence.

III. Fast Level Set Implementation

In this section, we present a fast level set implementation of the curve evolution process in (2) when the scene is composed of only the background Ω 0 and a single object region Ω 1. Extensions are then made to track multiple objects with different feature distributions in the next section.

To represent the background Ω_0 and the object region Ω_1, we use a level set function Φ which is negative in Ω_1 and positive in Ω_0. Based on this representation, we define two lists of neighboring pixels Lin and Lout of C1 as shown in Fig 6.
Formally, they are defined as:

$$\text{Lout}=\{X| \phi(x) > 0 \text{ and } \exists_y \epsilon N_1(x) \text{such that } \phi(y) < 0\}$$
$$\text{Lin}=\{X| \phi(x) < 0 \text{ and } \exists_y \epsilon N_1(x) \text{such that } \phi(y) > 0 \}$$

where N4(x) is a 4-connected discrete neighborhood of a pixel x with x itself removed. In conventional implementations of the level set method, an evolution PDE is solved either globally on the whole domain or locally in a narrow band to evolve the curve according to (2). Our fast level set implementation is based on the key observation that the implicitly represented curve C1 can be evolved at pixel resolution by simply switching the neighboring pixels between the two lists Lin and Lout.

For example, as we show in Fig., the curve C1 moves outward at pixel A and shrinks and splits at pixel B compared with the curve shown in Fig. This motion can be realized by simply switching pixel A from Lout to Lin, and pixel B from Lin to Lout. By doing this for all the pixels in Lin and Lout, the curve can be moved inward or outward for one pixel everywhere in one scan. Since the curve is still represented implicitly, topological changes can be handled automatically. With this idea, we can achieve level-set based curve evolution at pixel resolution and this is usually enough for many imaging applications. Next we present the details of our fast algorithm.

A. Basic Algorithm

The data structure used in our implementation is as follows:

* An array for the level set function ϕ;
* An array for the evolution speed F;
* Two bi-directionally linked lists of neighboring pixels: Lin and Lout.

Besides the inside and outside neighboring pixels contained in Lin and Lout, we call those pixels inside C1 but not in Lin as interior pixels and those pixels outside C1 but not in Lout as exterior pixels. For faster computation, we define Φ as follows:

$$\phi(x) = \begin{cases} 3 & if\ x\ is\ an\ exterior\ pixel \\ 1 & if\ x \in L_{out} \\ -1 & if\ x \in L_{in} \\ -3 & if\ x\ is\ an\ interior\ pixel \end{cases}$$

To switch pixels between Lin and Lout, we define two basic procedures on our data structure.

The procedure $switch\_in(x)$ for a pixel $x \in L_{out}$ moves the curve outward one pixel at x by switching it from Lout to Lin and adding all its neighboring exterior pixels to Lout. Formally this procedure is defined as follows:

DOI: 01.IJCOM.4.1.1

$switch\_in(x)$:

Step 1: Delete x from L_{out} and add it to L_{in}. Set $\phi(x) = -1$

Step 2: $\forall\, y \in N_4(x)$ satisfying $\phi(y) = 3$, add y to L_{out} and . Set $\phi(y) = 1$

Similarly, the $Switch\_out()$ procedure that moves the curve inward one pixel at $x \in L_{in}$ is defined as follows:

$Switch\_out(x)$:

Step 1: Delete x from L_{in} and add it to L_{out}. Set $\phi(x) = 1$

Step 2: $\forall\, y \in N_4(x)$ satisfying $\phi(y) = -3$, add y to L_{in} and . Set $\phi(y) = -1$

To track the object boundary, we compute the speed at all pixels in Lout and Lin and store their sign in the array F. We first scan through the list Lout and apply a $Switch\_in()$ procedure at a pixel if F = +1. After this scan, some of the pixels in Lin become interior pixels and they are deleted. We then scan through the list Lin and apply a $Switch\_out()$ procedure for a pixel with F = - 1. Similarly, exterior pixels in Lout are deleted after this scan. At the end of this iteration, a stopping condition is checked. If it is satisfied, we stop the evolution; otherwise, we continue this iterative process. In our implementation, the following stopping condition is used: Stopping Condition The curve evolution algorithm stops if either of the following conditions is satisfied:

(a) The speed at each neighboring pixel satisfies:

$$F(x) \leq 0 \quad \forall\, x\, \varepsilon\, L_{out}$$

$$F(x) \geq 0 \quad \forall\, x\, \varepsilon\, L_{in}$$

(b) A pre-specified maximum number of iterations is reached.

The condition in (4) is very intuitive in the sense of region competition. When the curve is on the object boundary, all the pixels in Lout are in the background and all the pixels in Lin are in the object region. When (4) is satisfied, they disagree with each other on which direction to move the curve and convergence is reached. When the data is noisy or there is clutter, regularization is necessary and (4) may not be always satisfied in the final curve. Thus part (b) of the condition is also necessary to stop the evolution.

The above algorithm can be applied to arbitrary speed fields and speeds up the evolution process in (2) dramatically compared with previous narrow band techniques based on solving the level set PDE. For the curve evolution equation in (2), we can achieve a further speedup by introducing a novel scheme that separates the evolution driven by the data dependent speed Fd and the smoothing speed Fs into two different cycles. In spirit, this idea is similar to the work in [10] which proposed a fast method to implement the Chan-Vese model [5] over the whole domain, but the two-cycle algorithm we present next is still based on updating the two linked lists Lin and Lout to evolve the implicitly represented curve.

B. Two cycle Algorithm

In the curve evolution equation in (2), the speed Fs for smoothness regularization is a function of the curvature, which is computationally quite expensive to evaluate generally. However when the level set function is chosen as the signed distance function, the curvature takes a simpler form and equals the Laplacian of ϕ. From the theory of scale-space [18], we know that evolution of a function according to its Laplacian is equivalent to Gaussian filtering the function. Motived by this observation, we propose to incorporate smoothness regularization using a Gaussian filtering process to further speed up our algorithm.

Let us denote an isotropic Gaussian filter of size NgXNg as G. To smooth the zero level set, we compute the response of Φ to the Gaussian filter only at pixels in the two lists Lin and Lout. Due to the need of maintaining our data structure, we do not take the output of the Gaussian filter directly. Only when the sign of the output is different from the original value of Φ at a pixel, we apply a $Switch\_in()$ or $Switch\_out()$ procedure to move the zero level set; otherwise, the original value of Φ is kept.

To combine this smoothing process with the evolution of the data dependent speed Fd, we propose a two-cycle algorithm. In the first cycle of our algorithm, we evolve the curve according to Fd using the fast algorithm. In the second cycle, we apply the Gaussian filtering process to incorporate smoothness regularization.

The details of this algorithm are listed in Table 1. Two factors make this two-cycle algorithm faster than curvature-based regularization. When the noise is low, we can reduce the parameter Ng or increase the parameter Na to reduce the percentage of computation allocated for smoothness regularization. The second reason is that we can implement the Gaussian filtering process with integer operations since we only care about the sign of its output.

To conclude, we will use the two-cycle algorithm in our level-set-based tracking. With this fast implementation, real-time tracking can be achieved.

IV. Tracking Multiple Objects

In this section, we extend our fast algorithm to track multiple objects with different feature distributions. For the representation of multiple object regions, we use two functions: one region indication function ψ and one level set function ϕ. This simple representation is motivated by the work in [8]. The region indication function is defined as follows:

$$\psi(x) = m, if\ x \in \Omega_m (m = 0,1, \ldots, M)$$

For the tracking of each frame, we assume the initial curves for all the object regions are separated by the background region, but arbitrary topology is allowed in the final tracking result. The level set function is negative inside all the object regions and positive in the background. For each object region , two lists of neighboring pixels and can be defined as we did for the case of single object region. Here the interior pixels are those pixels inside the object regions but not

DOI: 01.IJCOM.4.1.1

contained in any in, and exterior pixels are those pixels in the background but not in any . Similar to the two region case, the level set function is defined as:

TABLE I. THE TWO CYCLE FAST ALGORITHM

Step1 : Initialize the array ϕ, F_d the two lists L_{out} and L_{in}

Step 2 : (Cycle one): for $i = 1 : N_a$ do

−Compute the speed F_d for the pixel in L_{out} and L_{in};
−for each pixel $x \in L_{out}$ switch$_{in(x)}$ if $F_d(x) > 0$
−for each pixel $x \in L_{in}$, if $\forall\, y \in N(x), \phi(y) < 0$, delete x from L_{in}, set $\phi(x) = -3$
−foreach pixel $x \varepsilon L_{in}$ switch$_{out(x)}$ if $F_d(x) < 0$
- for each pixel $x \in L_{out}$ if $\forall\, y \in N(x), \phi(y) > 0$, delete x from L_{out} and set $\phi(x) = 3$
−Check the stopping condition. if it is satisfied, go to step 3: otherwise continue this cycle

Step 3 : (Cycle two): for $i = 1 : N_a$ do

− For every pixel x in L_{out} compute $G \otimes \phi(x)$. If $G \otimes \phi(x) < 0$ switch$_{in(x)}$;
−For each pixel $x \in L_{in}$ if $\forall\, y \in N(x), \phi(y) < 0$ delete x from L_{in} and set $\phi(x) = -3$
−For ecery pixel x in L_{in} compute $G \otimes \phi(x)$. If $G \otimes \phi(x) > 0$ switch$_{out(x)}$.
−For each pixel $x \in L_{out}$ if $\forall\, y \in N(x), \phi(y) > 0$, delete x from L_{out} and set $\phi(x) = 3$

Step 4: If the stopping condition is satisfied in cycle one, terminate the algorithm, otherwise, go back to step 2

$$\phi(x) = \begin{cases} 3 & \text{if } x \text{ is an exterior pixel} \\ 1 & \text{if } x \in L_{out}^m \text{ for any } 1 \le m \le M \\ -1 & \text{if } x \in L_{in}^m \text{ for any } 1 \le m \le M \\ -3 & \text{if } x \text{ is an interior pixel} \end{cases}$$

To evolve the M curves C1;C2; _ _ _ ; CM and keep track of all the regions, we evolve Φand Ψ simultaneously while keeping all the object regions from merging with each other.

This kind of topology control requirements can be realized easily in our fast algorithms by incorporating ideas in discrete topology [3, 11]. In the following, we introduce related concepts in the context of video tracking. For more detailed discussion about topological numbers and discrete topology, see [3].

Let $N_8(x)$ denote the 3 × 3 neighborhood centered at a pixel x but with x removed, and Ω_{obj} denote the union of all the object regions $\Omega_m (1 \le m \le M)$. The topological number of x with respect to Ω_{obj} is the number of 4-connected components in the set $\Omega_{obj} \cap N_8(x)$ and we denote it as $T_{obj}(x)$. The topological number of x with respect to the background region Ω_0 is the number of 8- connected components in the set $\Omega_0 \cap N_8(x)$ and we denote it as $T_{bg}(x)$. According to [3], the pixel x is a simple point if both = 1 and = 1. For a simple point, its removal or addition to will not change the topology of Ω_{obj}.

Based on the idea of topological numbers, a topology preserving level set method was developed in [11] which does not permit any splitting or merging of the curve. Here our requirement is more challenging. We want to allow each region $\Omega_m (1 \le m \le M)$ to have more than one connected components and they can merge and split at will if they belong to the same object region, but we also want to prohibit the merging of two connected components if they are from different object regions. To achieve this goal, we propose the concept of the relaxed topological number for a pixel $x \in L_{out}^m (1 \le m \le M)$ as follows:

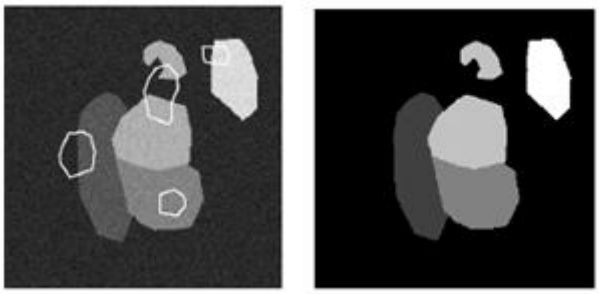
Definition For a pixel $x \in L_{out}^m (1 \le m \le M)$, if the number of object regions that intersect with $N_8(x)$ is a(x), its topological number with respect to Ω_{obj} is $T_{obj}(x)$, and its topological number with respect to Ω_0 is $T_{bg}(x)$, then the relaxed topological number of x is defined as:

$$T_r(x) = \min\,(\alpha(x), \max\,(T_{obj}(x), T_{bg}(x)))$$

If x is a simple point, then $T_r(x) = 1$ and its addition to an object region will not cause two different object regions to merge. If x is not a simple point but a(x) = 1, we still have $T_r(x)$ =1. This means that the addition of x to an object region will cause the merge of connected components from the same object region. For all the other cases, we have $T_r(x) > 1$, thus the relaxed topological number at a pixel can be used to prevent the merging of different object regions while allowing flexible topological changes within each object region. Based on this idea, we modify the $Switch\_in()$ and $Switch\_out()$ procedure to update ϕ and ψ to track the M object regions as we evolve the zero level set. The modified procedures are listed as follows:

switch_in(x) for $x \in L_{out}^m (1 \le m \le M)$.
step 1: Compute $T_r(x)$. If $T_r(x) = 1$, continue to step 2, otherwise exit the procedure.
Step 2: Delete x from L_{out}^m and add it to L_{in}^m.

DOI: 01.IJCOM.4.1.1



a b

Figure 4. (a) The original image and the initial curves (in white). One initial curve is used for each object region. (b) The segmentation result is shown in the region indication function

$$Set\ \phi(x) = -1\ and\ \psi(x) = m$$
$$Step\ 3: \forall\ y \epsilon N_4(x) satisfying\ \phi(y) = 3,$$
$$add\ y\ to\ L_{out}^m, and\ set \phi(y) = 1$$
$$switch\_out(x) for\ x \epsilon L_{in}^m (1 \le m \le M).$$
$$Step\ 1: Delete\ x\ from\ L_{in}^m\ and\ add\ it\ to\ L_{out}^m.\ Set\ \phi(x) = 1\ and\ \psi(x) = 0.$$
$$Step\ 2: \forall\ y \epsilon N_4(x)\ satisfying\ \phi(y) = -3, add\ y\ to\ L_{in}^m, and\ .Set\ \phi(y) = -1$$

Similar to the case of a single object region, we use a two-cycle algorithm to track multiple object regions. Compared with the algorithm used to track one object, we need to scan through 2M lists L_{in}^m and L_{out}^m (m = 1; 2; _ _ _;M) in each iteration of cycle one and two to track M objects.

After the two-cycle algorithm finishes, we perform a likelihood test for each pixel in $L_{out}^m = (1 \le m \le M)$ with $T_r(x) > 1$ and set its value in the region indication function as:

$$\psi(x) = \underset{m \epsilon S_m}{argmax}\ p(v(x)|\Omega_m)$$
$$S_M = \{1 \le m \le M | \Omega_m \cap N_4(x) \ne \emptyset\}$$

Here $P(.|\Omega_m)$ is the feature distribution for each object region used in (1). By performing this likelihood test, we can assign pixels enforced as background before to proper object regions and enable the final tracking result to have arbitrary topology, such as multiple object regions connecting with each other. With all the regions labeled, the final tracking result is saved in for the current frame and we move on to the next frame. In Fig. 4, we illustrate an image segmentation example using our algorithm. The noisy original image shown in Fig. 4(a) is composed of the background region and four object regions. We start with four initial curves to localize the four object regions of different intensities. With our method, successful segmentation is obtained and we show the region indication function in Fig. 4(b). This example demonstrates the power of our method to represent and localize multiple regions with the desired topology.

V. Result

We have presented a set of algorithms based on color histogram back projection which simplify the tracking of multiple moving objects within video sequences. The adaptive color histogram back projection algorithm uses both adaptive object models and adaptive blob detector thresholds. In addition, object position is tracked to enhance the performance of the algorithm. To achieve this we derive motion cues from existing back projection and thereby avoid complex computation of object velocities. An interactive tracking system CLICK-IT was implemented to demonstrate multi-object tracking for various surveillance-like and sports video sequences.

Tests on these sequences so far indicate that the system is quite robust, on average a tracking success rate of greater than 93% was achieved on sequences of lengths about 10 seconds. Robustness might be improved using a combination of color histograms and Kalman filters, at the expense of increased computational complexity.

A possible development of the system would be to "piggyback" onto a camera system with pan-tilt-zoom functions controlled remotely by a joystick. The tracker could be linked by a feedback loop to help keep objects of interest centered and highlighted. An obvious next step would be automated camera control.

References

[1] Akgül CB, Sankur B, Schmitt F, Yemez Y (2007) "Multivariate density-based 3D shape descriptors." Proceedings of shape modeling international, pp 3–12

[2] Ankerst M, Kastenmuller G, Kriegel H-P, Seidl T (1999) "3D Shape histograms for similarity search and classification in spatial databases." Proc. symposium on large spatial databases, pp 207–226

[3] Ansary TF, Vandeborre J-P, Mahmoudi S, Daoudi M (2004) "A bayesian framework for 3D models retrieval based on characteristic views." Proc. 2nd international symposium on 3D data processing, visualization, and transmission

[4] Atmosukarto I, Leow WK, Huang Z (2005) "Feature combination and relevance feedback for 3D model retrieval." Proceedings of the 11th international multimedia modelling conference, pp 334–339

[5] Barros J, French J, Martin W, Kelly P, Cannon M (1996) "Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval." Proc. SPIE conference 267, pp 392–403

[6] Basri R, Costa L, Geiger D, Jacobs D (1998) "Determining the similarity of deformable shapes." Vis Res 38:2365–2385

[7] Belongie S, Malik J, Puzicha J (2002) "Shape matching and object recognition using shape contexts." IEEE Trans Pattern Anal Mach Intell 24(4):509–522

[8] Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourrne PE (2000) "The protein data bank." Nucleic Acids Res 28:235–242

[9] Besl PJ, Jain RC (1985) "Three-dimensional object recognition." Comput Surv 17(1):75–145

[10] Bespalov D, Shokoufandeh A, Regli WC (2003) "Reeb graph based shape retrieval for CAD." Proc. ASME design engineering technical conferences, computers and information in engineering conference

[11] Bespalov D, ShokoufandehA, RegliWC, SunW(2003) "Scale-space representation of 3Dmodels and topological matching." Proc. solid modeling 2003, pp 208–215

DOI: 01.IJCOM.4.1.1

[12] Biasotti S, Giorgi D, Marini S, Spagnuolo M, Falcidieno B (2006) "A comparison framework for 3D object classification methods." Multimedia content representation, classification and security, proceedings international workshop, MRCS 2006. Lecture notes in computer science, vol 4105, pp 314–321

[13] Biasotti S, Marini S, Mortara M, Patané G (2003) "An overview of properties and efficacy of topological skeletons in shape modelling." Proc. shape modeling international 2003, pp 245–254

[14] Biasotti S, Marini S, Mortara M, Patané G, Spagnuolo M, Falcidieno B (2003) "3D shape matching through topological structures." Proc. discrete geometry for computer imagery, pp 194–203

[15] Bimbo AD, Pala P (2006) "Content-based retrieval of 3D models." ACM Trans Multi Commun App 2(1):20–43

[16] Bustos B, Keim D, Saupe D, Schreck T, Vrani´c D (2004) "Automatic selection and combination of descriptors for effective 3D similarity search." Proc. IEEE international workshop on multimedia content-based analysis and retrieval

[17] Bustos B, Keim D, Saupe D, Schreck T, Vrani´c D (2004) "Using entropy impurity for improved 3D object similarity search." Proc. IEEE international conference on multimedia and expo

[18] Bustos B, Keim DA, Saupe D, Shrec T, Vranic DV (2005) "Feature-based similarity search in 3D object databases." ACM Comput Surv 37:345–387

[19] Campbell RJ, Flynn PJ (2001) "A survey of free-form object representation and recognition techniques." Comput Vis Image Underst 81(2):166–210

[20] Cardone A, Gupta SK, Karnik M (2003) "A survey of shape similarity assessment algorithms for product design and manufacturing applications." J Comput Inf Science Eng 3:109–118

[21] Chen D-Y, Tian X-P, Shen Y-T, Ouhyoung M (2003) "On visual similarity based 3D model retrieval." Comput Graph Forum (EG 2003 Proceedings) 22(3):223–232

[22]Chua SJ, Jarvis R (1997) "Point signatures: a new representation for 3D object recognition." Int J Comput Vis 25(1):63–65

DOI: 01.IJCOM.4.1.1