OBJECT TRACKING IN REAL TIME VIDEO SEQUENCES USING A FAST LEVEL SET METHOD\(^1\)

BY

BOGDAN APOSTOL and VASILE MANTA

Abstract. We propose a method for determining the shape (2D outline of the object) and follow-up position (set of simple geometric transformations on position), using a level set based curve evolution and combining the benefits of using the pixel-wise posterior term with a fast level-set algorithm to approximate curve evolution. The pixel-wise posterior allows us to marginalize the model parameters at pixel level, and the fast level-set algorithm avoids the need of solving partial differential equations (PDEs). Our proposed implementation can accurately process a higher number of frames per second, bringing real-time performance on standard hardware systems.

Key words: object tracking, active contours, image segmentation, fast level set, PWP.

2000 Mathematics Subject Classification: 68U05, 65D18.

1. Introduction

Object tracking and shape adaptation, according to changes in the appearance of moving objects and/or camera movement, is an intensely studied topic of various applications in computer vision. The process of finding the contour and adapting the position of an object in the scene in video sequences is composed of image segmentation and pose geometric transformations.

The basic idea in active contour based segmentations (introduced by Kass, Witkins and Terzopoulos [3]) is to evolve a curve, under the influence of image data constraints, in order to separate the relevant objects from the background. We start with an initial approximation of the segmentation: a curve around the object. This moves in the normal direction and stops at the object

---

\(^1\) This is an extended version of the paper: Apostol B. and Manta V., Object Tracking in Real Time Video Sequences Using a Fast Level Set Method. Proc. ICSTC 2010, 31–36, 2010.
contour. Classical snake models involve an edge detector to stop the evolving curve at the border of the object. As shown in [2], the snake model has the following problems: it doesn't allow the contour to easily undergo topological changes, it lacks a probabilistic interpretation and can it be quite sensitive at initialization (the algorithm tends to get stuck in undesirable local minima).

We represent the contour of the object as the zero level set, an implicit function \( \phi \) defined in a higher dimensional space, usually mentioned as the level set function, and evolve the curve according to a partial differential equation (PDE). This approach has the advantage that the level set can separate and merge naturally during evolution, thus topological changes are handled automatically. The level set formulation also permits us the use of statistical region-based methods, thus there are less local minima in the image data and the segmentation schemes are far less sensitive to noise and to initialization varying.

A region based method models the foreground and the background membership probabilities. The standard method of Mumford and Shah is extended in [8] where a per pixels log likelihood is computed from the per region histogram. Another method is that of [1] where a per pixel posterior term is computed from the foreground and background aspect models. This method provides better result than [2], [8] as it allows us to marginalise out model parameters at a pixel level and allow the shape to change online.

In this paper, we combine the benefits of using the pixel-wise posterior term [1], as opposed to a likelihood, with the benefits of using an algorithm for the approximation of level-set-based curve evolution [6]. This, unlike [1], avoids the need to solve the PDE at every step and speeds up the algorithm.

Similar to [1], we define a probabilistic framework based on two aspect models, respectively the foreground model and background model. Our models are represented by RGB histograms using 32 bins per channel. We initialize the models using a user imputed region of interest. The values of the pixels inside the region of interest are used to build the foreground model, and the values of the pixels outside the region of interest are used to build the background model. The two initial distributions are then used to generate an initial segmentation, and then to rebuild the probabilistic model. This step is repeated until the shape converges. Similar to [1] we adapt the foreground and background models online, by using a linear model with learning rates.

The level-set implementation proposed in [1] is still slow on standard hardware, because the curve evolution process is based on the solution of certain partial differential equations (PDEs). This takes significant computational time for each of the steps mentioned above. In this paper we use the fast level set approximation [8], where two linked lists are kept that represent a narrow band of one pixel width, around the actual contour. The level set evolution is simply a switching mechanism between the two lists.
The outline of this paper is as follows: Section 2 describes the use of level-set segmentation and pixel-wise posteriors; Section 3 outlines the implementation details and shows the experimental results of the method both in real-time videos and post-processed image sequences and Section 4 concludes with a summary and discussion of further development possibilities.

2. Theoretical Considerations

2.1. Level Set Method

a) Fundamentals. The main idea of the level set methodology is to embed the propagating curve as the zero level of Lipchitz continuous functions $\phi$. Let $\{(x,y) \in \mathbb{R}^2 : \phi(x,y) = 0\}$ be the embedding function for the level set function. If we associate a velocity field $v$ and by limiting it to the curve we determine the speed of the curve, then at least for a moment time $t$, the evolution can be described with a Cauchy approach [7]:

$$\phi_t + v \cdot \nabla \phi = 0, \quad \phi(x,0) = \phi_0(x),$$

where: $\phi_0$ embeds the initial curve position. Enforcing the parameterised curve to be the zero level of function $\phi$ in eq. (1), independent of the time variable, so that $\phi(\gamma(s,t),t) = 0$, whatever $t \geq 0$, we can write:

$$\phi_t + \frac{\partial \gamma}{\partial t} \cdot \nabla \phi = 0,$$

where: $\frac{\partial \gamma}{\partial t}$ is the known dynamics of the curve. Extending $\frac{\partial \gamma}{\partial t}$ continuously throughout the entire domain will create such a field of velocities. Generally, the velocity field $v$ can be a function of position and other geometric quantities of the curve.

We can rewrite eq. (2) using the normal velocity:

$$\phi_t + v_n \frac{\nabla \phi}{|\nabla \phi|} = 0, \quad v_n = v \cdot \frac{\nabla u}{|\nabla u|}.$$

In the level set formulation, the surface integral of function $f$ along the zero level is defined by [7]:
where: $\delta(\phi)$ is the delta Dirac function.

If $f \equiv 1$, the result of this integral is the arc length for curves in two-dimensional spaces, the surface area in three-dimensional space and eq. (4) becomes:

$$\int_{\mathbb{R}^d} f(x) \delta(\phi) |\nabla \phi| \, dx.$$  

Volume integrals are defined as:

$$\int_{\mathbb{R}^d} H(\phi) |\nabla \phi| \, dx.$$  

where: $H$ is defined as the step function:

$$H(\phi) = \begin{cases} 1, & \text{for } x \geq 0 \\ 0, & \text{for } x < 0 \end{cases}$$

b) **Reshaping the level set function.** The level set function develops steep and smooth gradients, generating problems for numerical approximations, and it becomes necessary to reshape the level set function in a usable form, but without changing the zero level location. This means reshaping the level set around the interface and leaving the interface as is.

A method of remodelling is reinitializing the distance by evolving the PDE function (partial equations) to a steady state [7]:

$$\phi_t + \text{sgn}(\phi_0) \left( |\nabla \phi| - 1 \right) = 0, \quad \phi(x, t = 0) = \phi_0(x)$$

If it evolves towards a steady state solution on the computational domain, the solution of $\phi$ becomes the signed distance transform function for the interface $\{\phi_0 = 0\}$. In the region where $\phi_0$ is positive, $\phi_t < 0$ when $|\nabla \phi| > 1$ and therefore, the value of $\phi$ will decrease and thus $|\nabla \phi|$ will become closer to 1. We note that $\phi_t = 0$ when $\phi_0 = 0$, because $\text{sgn}(\phi_0) = 0$. 


Standard level set uses a common numerical method, which implies the reshaping the distance form $\phi$ (Fig. 1) with a higher order more accurate method for a very short time interval, so that a small band around the $\phi=0$ interface. This method is called the distance transform function (Fig. 2) and the values of $\phi$ inside the band become the values of the distance transform.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1.png}
\caption{Reshaping the function $\phi$.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{Distance transform method: $a - \phi = 0$ interface; $b$ – Value of each pixel as the distance to the nearest pixel on interface.}
\end{figure}

### 2.2. Fast Approximation of Level Set

a) Representation of curve evolution. The level set method represents the interface as the zero level of a function $\phi$, defined over a regular grid $D$ of $k$ dimensions. We assume the grid is sampled uniformly and the default sampling is of uniform distance 1. The coordinates of a point in the grid are given as $x = \{x_1, x_2, \ldots, x_k\}$. We define the list of inside neighbouring grid points in the band $L_{\text{inside}}$ and outside neighbouring grid points in the band $L_{\text{outside}}$ for the object regions as follows:
Bogdan Apostol and Vasile Manta

\[(9)\]

\[\begin{align*}
L_{\text{inside}} &= \{ x, x \in \Omega \text{ and } \exists y \in N(x) \text{ such that } y \in D \setminus \Omega \}, \\
L_{\text{outside}} &= \{ x, x \in D \setminus \Omega \text{ and } \exists y \in N(x) \text{ such that } y \in \Omega \},
\end{align*}\]

where: \( N(x) \) is a discrete neighbourhood of \( x \).

The interface moves inward or outward and can be split into two curves, as the value of \( \phi \) changes from positive to negative. To move the curve in one or the other direction one needs to solve the PDE equations, which are time and resource consuming.

The idea that [5] proposes is to only take into account the border between the object and the background area, and to achieve the same result if we use the relation between the two lists, \( L_{\text{inside}} \) and \( L_{\text{outside}} \). This achieves the movement of the curve using minimal computation, by simply moving a grid point from one list to another, \( L_{\text{inside}} \) to \( L_{\text{outside}} \) if the curves move outward and \( L_{\text{outside}} \) to \( L_{\text{inside}} \) if the curves move inward.

The level set function values are locally approximated to a signed distance transform:

\[(10)\]

\[\hat{\phi}(x) = \begin{cases} 
3, & \text{for } x \text{ outside } C, x \notin L_{\text{outside}} \\
1, & \text{for } x \in L_{\text{outside}} \\
-1, & \text{for } x \in L_{\text{inside}} \\
-3, & \text{for } x \text{ inside } C, x \notin L_{\text{inside}}
\end{cases}\]

and the evolution speed \( F \) is represented by an integer-value array:

![Moving the curve by moving points from \( L_{\text{inside}} \) to \( L_{\text{outside}} \) and from \( L_{\text{outside}} \) to \( L_{\text{inside}} \)](image)
To keep the smoothness of the curve, [6] proposes a two step algorithm based on the fact that the evolution speed $F$ can be split in two components. The first one, $F_d$, is a data-dependent speed, which is a function of the image data and depends on up to the first order geometric properties of the curve, and the second one, $F_{int}$, is a smoothing regularization speed, which is proportional to the mean curvature. The two component speeds must keep:

$$F(x) = F_d(x) + F_{int}(x).$$

The data-dependent speed $F_d$ is represented by the integer-value array $\hat{F}_d$, and the smoothing speed $\hat{F}_{int}$ is represented by the integer-value array, using the sign representation in eq. (11).

b) Fast two-cycle algorithm. This subsection will describe the two steps of the algorithm corresponding to evolving the curve using the sign of the data-dependent speed $\hat{F}_d$ and respectively evolving the curve using the sign of smoothing speed $\hat{F}_{int}$.

Step 1: initializes the arrays $\hat{F}_d$, $\hat{F}_{int}$ and the lists $L_{inside}$ and $L_{outside}$, using eqs. (10),...,(12).

Step 2: represents cycle one, data-dependent speed evolution:

- compute $F_d$ in each point stored in the two lists, $L_{inside}$ and $L_{outside}$ and store it’s sign in $\hat{F}_d$;
- evolve the curve outward and copy each point $x \in L_{outside}$ in $L_{inside}$ if $\hat{F}_d(x) > 0$ and then eliminate duplicates in $L_{inside}$;
- evolve the curve inward and copy each point $x \in L_{inside}$ in $L_{outside}$ if $\hat{F}_d(x) < 0$ and then eliminate duplicates in $L_{outside}$;
- check stopping condition and if satisfied continue with step 3, otherwise restart step 2.

Step 3: represents cycle two, smoothing speed evolution using Gaussian filtering:
− compute $F_{\text{int}}$ in each point stored in the two lists, $L_{\text{inside}}$ and $L_{\text{outside}}$ and store it’s sign in $\hat{F}_{\text{int}}$;
− evolve the curve outward and copy each point $x \in L_{\text{outside}}$ in $L_{\text{inside}}$ if $\hat{F}_{\text{int}}(x) > 0$ and then eliminate duplicates in $L_{\text{inside}}$;
− evolve the curve inward and copy each point $x \in L_{\text{inside}}$ in $L_{\text{outside}}$ if $\hat{F}_{\text{int}}(x) < 0$ and then eliminate duplicates in $L_{\text{outside}}$;

Step 4: if stopping condition from step 2 not satisfied then return to step 2.

2.3. Fast implementation of PWP

Traditional region based segmentations compute the overall likelihood $P(I|M)$ as the product of the pixel-wise likelihoods functions in all grid points $P(I|M) = \prod_{i=1}^{N} P(I(x_{i})|M_{i})$. Using the notations:
− $x_{i} = (x_{i}, y_{i})$ a pixel in the image,
− $M = \{M_{f}, M_{b}\}$ foreground model or background model,
− $I(x_{i})$ the image pixel value,
− $P(y|M_{f})$ foreground model over pixel values $y$ (RGB value),
− $P(y|M_{b})$ background model over pixel values $y$ (RGB value),
we define the foreground probability $P_{f}$ of a pixel, and the background probability $P_{b}$ as in [4]:

$$
P_{f} = \frac{P(y|M_{f})}{\eta_{f} P(y|M_{f}) + \eta_{b} P(y|M_{b})},$$

(13)

$$
P_{b} = \frac{P(y|M_{b})}{\eta_{f} P(y|M_{f}) + \eta_{b} P(y|M_{b})},$$

where: $\eta_{f} = \sum_{i=1}^{N} H_{\varepsilon}\left(\phi(x_{i})\right)$ and $\eta_{b} = \sum_{i=1}^{N} 1 - H_{\varepsilon}\left(\phi(x_{i})\right)$, $H_{\varepsilon}$ − the blurred Heaviside step function used in most level-set methods.

Next we define the data-dependent speed $F_{d}$ from eq. (12) as the difference between $P_{f}$ and $P_{b}$:
(14) \[ F_d(x) = P_f(x) - P_b(x), \]

and

(15) \[ \hat{F}_d(x) \geq 0, \forall x \in L_{\text{inside}} \]
\[ \hat{F}_d(x) \leq 0, \forall x \in L_{\text{outside}} \]

The smoothing speed \( \hat{F}_{\text{in}} \) is defined as follows for all the boundary points in the \( L_{\text{outside}} \) list:

(16) \[ \hat{F}_{\text{in}1}(x) = \begin{cases} 1, & \text{if } G \ast H_x > \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \]

where: \( H_x \) is the blurred Heaviside step function, \( G \) – the Gaussian filter and \( \ast \) – the convolution operation.

The smoothing speed \( \hat{F}_{\text{in}} \) is defined as follows for all the boundary points in the \( L_{\text{inside}} \) list:

(17) \[ \hat{F}_{\text{in}1}(x) = \begin{cases} 1, & \text{if } G \ast H_x < \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \]

2.4. Visual Tracking

As it is shown in [1] it is possible to treat the tracking problem directly in a segmentation framework. For visual tracking of the object we will use a general geometric model, the notation \( x_0 = [x_0, y_0]^T \) for pixel image position before applying the geometric transformations, and \( x = [x, y]^T \) for pixel image position after applying the geometric transformations.

Using homogeneous coordinates we can describe a general model for possible transformations in an image:

(18) \[ x = T (t_x, t_y) \cdot R (\Theta) \cdot S (s_x, s_y) \cdot x_0, \]
where: \( T(t_x,t_y) \) is the translation in both \( x \) and \( y \) directions, \( R(\Theta) \) – the rotation of \( \Theta \) angle and \( S(s_x,s_y) \) is the scaling according to the two axes \( x \) and \( y \).

Detailing the model in eq. (18):

\[
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\Theta) & \sin(\Theta) & 0 \\ -\sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}
\]

In this paper we will use the unit matrix for the scale and rotation matrixes, because they are automatically handled by adapting the curve on-line. Thus, we simplify eq. (19) and define the model as:

\[
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}
\]

and \( p = \{t_x,t_y\} \) the positioning parameters of the object.

Differenting the energy function, as shown by [6], regarding the positioning parameters \( p_j \), we evolve the curve in a space defined by the positioning parameters:

\[
\frac{\partial P(\phi|\Omega)}{\partial p_i} = P(\phi|\Omega) \sum \frac{P_f - P_b}{H_f(\phi)P_f + (1 - H_f(\phi))P_b} \frac{\partial H_f(\phi)}{\partial p_i},
\]

where:

\[
\frac{\partial H_f(\phi(x_i,y_i))}{\partial p_i} = \frac{\partial H_f}{\partial p_i} \left( \frac{\partial \phi}{\partial x_i} \frac{\partial x_i}{\partial p_i} + \frac{\partial \phi}{\partial y_i} \frac{\partial y_i}{\partial p_i} \right) = \delta(X)(\phi) \left[ \frac{\partial \phi}{\partial x_i} \frac{\partial x_i}{\partial p_i} + \frac{\partial \phi}{\partial y_i} \frac{\partial y_i}{\partial p_i} \right] \text{ and } i = \{1,2\}.
\]

2.5. Online Learning

The online adaptation of the histogram models is done using the linear learning model proposed by [1]:
Thus, if the learning rate value is small, even if the current pixel distribution does introduce errors, they will be corrected by previous histogram models.

3. Implementation and Experimental Results

The application described in this paper has been tested mostly on live video sequences, but also on a set of recorded sequences and then processed frame by frame. These sequences contain: objects which show rapid movements, posture changes regarding the video camera, camera moving towards and outwards, scene illumination variations and image noise.

This section will present the implementation details and some of the qualities of the method, such as robustness, flexibility in tracking different kinds of objects, resilience to noise and high real time performance on a standard hardware system. We will also show the robustness of the implementation to changes of appearance and camera angle. Finally we will make a comparative study with the PWP implementation proposed by [1], varying the segmentation parameters, the limit number of iterations per frame, and the online learning rates.

The proposed implementation was tested on computer generated video sequences (three dimensional computer scenes) and on camera feed images. In testing on video sequences obtained by real-time camera feed, we used a Windows Vista compatible web camera, which uses the USB 2.0 connectivity. In this section from now on when we refer to the video source used in experiments, we refer to a Logitech QuickCam Pro 9000, which captures images at a resolution of 640x480, and a capture rate of 15 frames per second. The application was written in C++ programming language and uses a managed user interface in C# to adjust the parameters and register the results. It was tested on an Intel Core2Duo 2.16 GHz machine with an ATI Radeon 3470 video card.

![Fig. 3 – Foreground and background aspect models.](image-url)
The aspect models are computed using the total area of the image and not restricted only on the region of interest, as the curve is. Thus for the foreground histogram model all the pixels that have a positive level-set value are taken in account. For the background histogram model we take in account both the negative level-set value pixels and the pixels outside the region of interest. This is shown in Fig. 3.

Unless otherwise specified, we use the following parameter values for the learning rates of the two histogram models: $\alpha_f = 0.001$ and $\alpha_b = 0.002$.

First we present a qualitative evaluation for two image segmentations, Fig. 4. The inner surface of the contour is displayed in magenta color and is overlaid over the real model in the image. The first set of frames (Figs. 4a, 4b, 4c) represent the curve evolution until it reaches the real mug contour. For this experiment we choose a 100x100px region of interest and an initial circle contour of 80px diameter that was placed in the middle of the region of interest. In the second set of frames (Figs. 4d, 4e, 4f) we set the curve evolution for segmented a more complicated object as a hand. For this experiment we choose a 130x130px region of interest and an initial circle contour of 100px diameter that was placed in the middle of the region of interest.

![Fig. 4 – Curve evolution in image segmentation. a – iteration 10, b – iteration 70, c – iteration 120, d – iteration 10, e – iteration 50, f – iteration 360.](image)

Note that although the hand color distribution is close to the color of the background the contour does not evolve beyond the real edges of the hand figure. We also see the benefits of using a fast level-set implementation as we
can observe the quick merging of the contour in both cases, creating a complete and stable contour over the edges of the objects. Because we use a two list band representation ($L_{\text{inside}}$ and $L_{\text{outside}}$) isolated contours cannot appear as in the classical level-set implementation, where isolated regions can build up and pass through the zero level, thus creating isolated contours.

In Fig. 5 we selected a few frames obtained by real-time processing of a webcam image sequence. Note the contour evolution and on-line adaptation in the sequence of finding and tracking the human face, and that in time parts that were not segmented initially become part of the contour. Also, in the mug tracking sequence we see how the algorithm deals with scale adaptation (between (Figs. 5b and 5c)).

![Fig. 5 – Visual tracking and contour evolution for a real scene:](image)

- a, b, c – tracking the map; d, e, f – tracking human face.

4. Conclusions

In this paper, we have proposed a method of combining the benefits of using the pixel-wise posterior term, by evolving the curve depending on two Bayesian models, with the benefits of a fast-level-set based algorithm, that does not require us to solve the partial differential equations but rather approximate it. The pixel-wise-posterior term allows us to marginalize model parameters at pixel level, and the fast-level-set implementation permits us to evolve the curve by simply moving a pixel from one list to another. Thus, we obtain a highly accurate algorithm that can easily process more than 20 frames/sec, using a
web-camera live feed on a standard hardware system. Further algorithm development primarily involves increasing the processing by running jobs in parallel and by implementing the jobs directly on the Graphics Processing Unit.

Received: November 8, 2010

“Gheorghe Asachi” Technical University of Iași,
Department of Computer Engineering
and Applied Informatics
e-mails: vmanta@cs.tuiasi.ro
apostol_bgdn@yahoo.com

REFERENCES


URMĂRIREA OBIECTELOR ÎN TIMP REAL ÎN SECVENȚE VIDEO
FOLOSIND METODA DE SEGMENTARE RAPIDĂ DE EVOLUȚIE A FRONTURILOR

(Rezumat)

Această lucrare propune o metodă de determinare a formei (conturul bidimensional al unui obiect) și de urmărire a poziției (transformări simple ale posturii), folosind o evoluție a curbei definită de o funcție mulțime de nivele. Combinând avantajele folosirii unui termen probabilitate posterioră a fiecărui pixel, care ne permite
să marginalizăm parametrii modelului la nivel de pixel, cu un algoritm mulțime de nivele rapid care aproximează evoluția curbei fără a mai fi necesară rezolvarea ecuațiilor diferențiale parțiale. Implementarea propusă de noi poate procesa cu succes un număr de peste 20 de frame-uri/sec, aducând astfel performanță și acuratețe în timp real pe sisteme hardware standard. Rezultatele experimentale sunt prezentate în urma testării algoritmului pe un set de secvențe video filmate în timp real, cât și pe secvențe înregistrate sau care prezintă zgomot. Se arată invariabilitate algoritmului la schimbări rapide ale posturii camerei sau a obiectului urmărit, iluminarea scenei sau introducerea zgomotului în imagini.