Fast two-cycle curve evolution with narrow perception of background for object tracking and contour refinement

Yaochen Li, Yuanqi Su, Yuehu Liu

Abstract

The problem of object contour tracking in image sequences remains challenging, especially those with cluttered backgrounds. In this paper, the fast two-cycle level set method with narrow perception of background (FTCNB) is proposed to extract the foreground objects, e.g., vehicles from road image sequences. The curve evolution of the level set method is implemented by computing the signs of region competition terms on two linked lists of contour pixels rather than by solving partial differential equations (PDEs). The curve evolution process mainly consists of two cycles: one cycle for contour pixel evolution and a second cycle for contour pixel smoothness. Based on the curve evolution process, we introduce two tracking stages for the FTCNB method. For coarse tracking stage, the speed function is defined by region competition term combining color and texture features. For contour refinement stage which requires higher tracking accuracy, the likelihood models of the Maximum a posterior (MAP) expressions are incorporated for the speed function. Both the tracking and refinement stages utilize the fast two-cycle curve evolution process with the narrow perception of background regions. With these definitions, we conduct extensive experiments and comparisons for the proposed method. The comparisons with other baseline methods well demonstrate the effectiveness of our work.

© 2016 Elsevier B.V. All rights reserved.
CASES. MORE LITERATURES FOR OBJECT DETECTION AND TRACKING IN IMAGES AND VIDEOS ARE DESCRIBED IN [9–11].

Level set method is a powerful tool [12–15] for object tracking in image sequences due to its flexibility with respect to topological changes of the contours. The basic concept of this method assumes the boundary curve to be the zero level set of a function defined over the entire image domain. The boundary curve is then evolved by the predefined speed function. Conventionally, the level set implementation of the foreground curve evolution is based on the solution of certain partial differential equations (PDEs). However, the computational complexity of this PDE-based solution has limited the use of the level set method.

Two different approaches for the curve evolution of level set method have been examined by researchers. In the first approach, the update of the level set function is implemented globally over the entire image domain. The Chan–Vese (CV) model [16] is a typical example. The CV model applies the Mumford–Shah function [18] into the level set framework, which gives a piecewise constant description of an image. The curve evolution is driven by an energy function and incorporates a “fitting” term. Originally, the CV model is applied to grayscale images. Chan et al. [17] extend the traditional scalar-field CV model into a vector-field representation. Nevertheless, this model still ignores the consistency of spatial structure of the multichannel images. Gibou et al. [20] separate the CV model into two different cycles and connected the level set algorithm with K-means, as well as nonlinear diffusion preprocessing. The solving of the CV model using this method updates the level set function over the whole image grid.

The second approach focuses on the local evolution of the curve, and only solves the level set equations around the neighborhood of the zero level set. Peng et al. [19] propose a method to reinitialize the Hamilton–Jacobi equation for a few steps for each iteration of level set curve evolution. The level set curve evolution is performed on a localized region around the contours. Shi et al. [21] propose an approximation method of the level-set-based curve evolution without solving PDEs. The curve evolution is implemented by a switching mechanism between two linked lists of boundary pixels. The curve evolution process consisted of two cycles: one for data dependent terms and another for smoothness regularization. Although the curve evolution in Shi’s method is defined locally around the zero level set, the extraction of the foreground and background features is based on the global image domain. As a result, the foreground and background regions are confined to possess unique color and texture information in the experimental datasets, e.g. grayscale image sequences with static backgrounds. This has limited its usability. The method we propose in this paper falls into the second category.

In the present paper, we propose the fast two-cycle level set method with narrow perception of background (FTCNB) for road image sequences with dynamic background. This method consists of two tracking stages: (1) the object tracking stage and (2) the contour refinement stage. In the first stage, our algorithm is implemented based on the speed function that combines mid-level color and texture features. The object tracking is fast in speed and has adequate tracking accuracy. In the second stage, a maximum a posterior (MAP) expression is defined to infer the speed function. This is necessary for the generation of high-accuracy tracking contours. Both of these stages utilize a fast two-cycle curve evolution process with the narrow perception of background regions. The curve evolution process mainly consists of two cycles. The first cycle is designed for contour pixel evolution, while the second cycle is for contour pixel smoothness. The flow diagram of the proposed method is shown in Fig. 1. The main contributions of this paper are summarized as follows:

- The computations of the level set speed functions: In the object tracking stage, the speed function is defined by the region competition term that combines color and texture descriptors. The mid-level visual cues such as superpixels are applied to generate the color descriptors. The edge histogram descriptor is utilized as the texture descriptor. To obtain more accurate contours, the speed function is defined by the likelihood models of MAP expressions in the contour refinement stage.
- FTCNB level set method for outdoor image sequences: In both the object tracking and contour refinement stages, the proposed algorithms confine the perception of background to the close regions near the foreground contour. The two-cycle curve evolution algorithms are then implemented to achieve the resulted contours.
- Combination of both spatial and temporal information from the image sequences: The mid and low-level spatial cues are utilized in the object tracking and contour refinement stages, respectively. The temporal information between frames is considered for the contour refinement stage, i.e. the optical flows.

This paper builds upon and extends our previous work in [22], with a more detailed description of the algorithms and additional evaluation results. The remaining parts of this paper are organized as follows: Section 2 shows the object tracking stage of FTCNB method for coarse tracking. The data structure of the FTCNB level set method is introduced firstly. Motivated by the energy function, the speed function is defined and computed. The curve evolution process is then implemented. Section 3 shows

![Diagram](image_url)
the contour refinement stage of FTCNB method, where the new speed function is introduced based on the likelihood models of MAP. The optical flow map is applied as an assistant to compute the likelihood models. Section 4 presents the extensive experiments and comparisons. Section 5 lists the conclusion and future works.

2. Object tracking stage of FTCNB method

In this section, we introduce the object tracking stage of the FTCNB method. The data structure of the FTCNB method basically consists of the level set kernel matrix, speed function and contour pixel lists. The kernel matrix is applied to encode each image pixel to an integer value in the range of \([-3,3]\). Motivated by the data fidelity terms of the energy functions, the speed function is defined by the region competition terms on the contour pixel lists. The feature pools to support the computation of the speed function are updated for each frame. Based on these definitions, the fast two-cycle curve evolution can be implemented.

2.1. Data structure of the FTCNB method

Before we propose the curve evolution algorithms, the data structure of our method is defined as follows:

(a) an integer array \( \phi \) for the level set kernel matrix;
(b) an integer array \( F \) for the speed function;
(c) two linked lists of pixels that are adjacent to the evolving curve \( C \), denoted by \( L_{in} \) and \( L_{out} \).

The pixel lists \( L_{in} \) and \( L_{out} \) are defined for the boundary curve \( C \):

\[
L_{in} = \{ \mathbf{x} \in \Omega \text{ and } \exists \mathbf{y} \in N(\mathbf{x}) \text{ s. t. } \mathbf{y} \in D(\Omega) \},
\]

\[
L_{out} = \{ \mathbf{x} \in \Omega \text{ and } \exists \mathbf{y} \in N(\mathbf{x}) \text{ s. t. } \mathbf{y} \in \partial \Omega \},
\]

where \( \Omega \) is the foreground domain, i.e. the domain of the object being tracked. \( D \) is the image domain. \( \partial \Omega \) is the background domain, i.e. the domain outside the tracked object. \( N(\mathbf{x}) \) denotes the discrete neighborhood pixels of \( \mathbf{x} \) from the theory of digital topology [23].

With the assistance of the kernel matrix \( \phi \), we encode each image pixel to an integer value in the range of \([-3,3]\). The interior pixels correspond to the pixels inside \( C \) but not in \( L_{in} \), and the exterior pixels correspond to the pixels outside \( C \) but not in \( L_{out} \).

When the pixels in \( L_{in} \) and \( L_{out} \) are close in terms of color and texture information, we allow ambiguous pixels to exist between \( L_{in} \) and \( L_{out} \) as buffer area. The buffer area is constrained to a maximum width of \( \omega_{bc} \). The level set kernel matrix is defined by

\[
\phi(\mathbf{x}) = \begin{cases} 
3 & \text{if } \mathbf{x} \text{ is an exterior pixel} \\
1 & \text{if } \mathbf{x} \in L_{out} \\
0 & \text{if } \mathbf{x} \text{ is an ambiguous pixel} \\
-1 & \text{if } \mathbf{x} \in L_{in} \\
-3 & \text{if } \mathbf{x} \text{ is an interior pixel}.
\end{cases}
\]

Fig. 2(a) exhibits the update of kernel matrix \( \phi \) when the boundary curve evolves from time \( t \) to \( t+1 \). Fig. 2(b) visualizes the kernel matrix \( \phi \) in 3D domain at time \( t \).

2.2. Energy function definition

To define the energy function, the image domain can be partitioned into 3 parts: (1) the foreground region, (2) the close background region with narrow width of pixels, and (3) the far background region. The region partition example is shown in Fig. 3 (c). Motivated by the Mumford–Shah function [18] and the level set function based on it [20], we define the energy function \( E_1 \) as follows:

\[
E_1 = \mu \int_{\Omega} \| \nabla H(\phi) \| d\Omega - \lambda_1 \sum_{m=1}^{M} \int_{\Omega_m} \log p(\mathbf{x} | \Omega_m) H(\phi) d\Omega_m - \lambda_2 \sum_{b=1}^{B} \int_{\Omega_b} \log p(\mathbf{x} | \Omega_b)(1 - H(\phi)) d\Omega_b - \lambda_3 \log p(\mathbf{x} | \Omega_{bc})(1 - H(\phi)),
\]

where \( E_1, E_2 \) and \( E_3 \) are the data fidelity terms of foreground, close background and far background regions respectively. \( E_1 \) corresponds to smoothness regularization and is proportional to the length of all curves. \( M \) denotes the number of foreground regions. \( \Omega_m \) is the \( m \)-th foreground region and \( \Omega_b \) is the \( b \)-th close background region around \( \Omega_{bc} \) with width of \( \omega_{bc} \). \( \Omega_{bc} \) is defined as follows:

![Image](image_url)
\( \Omega_b = \left\{ x \in \Omega_m \mid \min_{y \in \Omega_m} \| x - y \| \leq \omega \right\}. \)

\( \Omega_b \) is the far background region (background region eliminating \( \Omega_b \)). \( \mu, \lambda_1, \lambda_2 \) and \( \lambda_3 \) are constant, non-negative coefficients. The ratio of the foreground domain to the image domain is set for both \( \lambda_1 \) and \( \lambda_2 \). The ratio of the background domain to the image domain is set for \( \lambda_3 \). \( \mu \) is set to 1 to make the smoothness weight higher. \( p(\cdot) \) is the posterior probability. \( f(x) \) is the feature vector defined at pixel location \( x \). \( H(\cdot) \) is the Heaviside distribution. For each component in \( f \), \( H(f(x)) \) is defined as follows:

\[
H(f(x)) = \begin{cases} 
0 & \text{if } f(x) \leq 0 \\
1 & \text{if } f(x) > 0.
\end{cases}
\]

2.3. Speed function definition

In this work, the energy function \( E_i \) is not designed for minimization but for the definition of speed function. The speed function determines the moving directions of the boundary pixels, which can be inferred by the data fidelity terms of the energy function. To make the curve evolution fast and more accurate, the perception of the background is constrained to the close background region adjacent to the foreground region when computing the speed functions. According to the data fidelity terms \( E_{d1} \) and \( E_{d2} \), the speed function \( F_{i} \) is defined by the region competition term [24] between foreground and close background regions at time \( t \):

\[
F_{i}^{c} = \log \frac{p(f(x)|\Omega_{m}^{f})}{p(f(x)|\Omega_{m}^{c})} = \log \frac{p(f_{i}(x)|\Omega_{m}^{f})p(f_{i}(x)|\Omega_{m}^{c})}{p(f(x)|\Omega_{m}^{c})p(f_{i}(x)|\Omega_{m}^{c})},
\]

where \( f_{i}(x) \) and \( f(x) \) are color and texture feature vectors at pixel location \( x \) respectively. \( \Omega_{m}^{f} \) and \( \Omega_{m}^{c} \) are the foreground and close background regions at time \( t - 1 \). \( f(x) \) can be computed either based on superpixels or K-means clustering in CIE-LAB color space. We choose the superpixel based approach. The superpixels are segmented by Mori’s method [25]. Fig. 3(a) shows examples of images segmented into superpixels of 32, 64, and 128 size pixels (from top-left to bottom-right in each image). In our experiments, the superpixel segmentation of 64 pixel size is preferred for it is fit for most of the experimental images. Moreover, each single superpixel keeps uniform color and texture information under this size. The initial boundary curves are extracted from the target-background saliency map generated from the user strokes, as depicted in Fig. 3(b). The superpixels on the stroke are utilized to define the foreground feature seeds in histograms. The Bhattacharyya distance of the histogram in each superpixel is then computed. We then set an empirical threshold to generate the saliency map. Fig. 3(c) shows the feature definitions of the different image regions, i.e. foreground, close background and far background regions.

We define \( C_{T}^{-1} \) and \( C_{b}^{-1} \) as the foreground and close background color feature pools at time \( t - 1 \). These feature pools are composed of the superpixels in foreground and close background regions of the previous frame. For each superpixel in the feature pools, a Gaussian distribution is constructed according to its mean color and variance.
where $S_{t-1}^{-1}(i)$ ($S_{b}^{-1}(i)$) denotes the $i_{th}$ Gaussian distribution in $C_{t-1}^{-1}$ ($C_{b}^{-1}$).

The K-means clustering of edge histogram descriptors [26] is applied to obtain the texture feature pools of foreground and close background: $T_{f}$ and $T_{b}$. The texture probabilities are defined as follows:

$$p\left( f_{t}(x) | Q_{t-1}^{-1} \right) = \sup_{S_{t-1}^{-1}(i) \in C_{t-1}^{-1}} \left\{ p\left( f_{t}(x) | S_{t-1}^{-1}(i) \right) \right\} .$$

$$p\left( f_{t}(x) | Q_{t-1}^{-1} \right) = \sup_{S_{b}^{-1}(i) \in C_{b}^{-1}} \left\{ p\left( f_{t}(x) | S_{b}^{-1}(i) \right) \right\} .$$

(7)

where $f_{t}(x)$ is the edge histogram descriptor centered at $x$, $T_{t-1}^{-1}$ and $T_{b}^{-1}$ are the texture feature pools at time $t-1$. $M_{t-1}^{-1}(i)$ and $M_{b}^{-1}(i)$ are the clustering means of the texture feature pools. $r(\bullet)$ is the average correlation coefficient [27] in the range of $[0, 1]$. The exponential distribution based on $r(\bullet)$ is defined as a probability, which is also in the range of $[0, 1]$.

2.4. Curve evolution of the object tracking stage

In the FTCNB level set method, the curve evolution is determined by the sign of $F_{t}^{d}$ rather than by solving PDEs. $switchIn()$ and $switchOut()$ functions are defined based on $F_{t}^{d}$ to implement the curve evolution. The $switchIn()$ function is utilized if a pixel in $L_{\text{out}}$ has a greater probability of belonging to the foreground than to the background. We delete this pixel from $L_{\text{out}}$ and add it to $L_{\text{in}}$ or to adjacent ambiguous pixels. The nearest exterior pixel is added to $L_{\text{out}}$. The $switchOut()$ function is utilized if a pixel in $L_{\text{in}}$ has a greater probability of belonging to the background than to the foreground. We delete it from $L_{\text{in}}$ and add to $L_{\text{out}}$ or adjacent ambiguous pixels. The nearest interior pixel is added to $L_{\text{in}}$.

Algorithm 1. Fast two-cycle curve evolution.

Cycle 1:
1. Compute the speed function $F_{t}^{d}$ for all the pixels in $L_{\text{out}}$ and $L_{\text{in}}$;
2. For each pixel $x \in L_{\text{out}}$, we $switchIn(x)$ if $F_{t}^{d}(x) > 0$; For each pixel $x \in L_{\text{in}}$, we $switchOut(x)$ if $F_{t}^{d}(x) < 0$;
3. Inspect each ambiguous pixel with an edge-to-center order: add it to $L_{\text{in}}$ if $F_{t}^{d}(x) > 0$; add it to $L_{\text{out}}$ if $F_{t}^{d}(x) < 0$; $\phi(x)$ is updated;
4. If a pixel $x \in L_{\text{in}}$ satisfies all the neighboring pixels in $\phi$ are non-positive, delete it from $L_{\text{in}}$ and set $\phi(x) = -3$; If a pixel $x \in L_{\text{out}}$ satisfies all the neighboring pixels in $\phi$ are non-negative, delete it from $L_{\text{out}}$ and set $\phi(x) = 3$;
5. Check if the stopping condition satisfied; if so, proceed to Cycle 2; otherwise, continue Cycle 1.

Cycle 2:
1. For each pixel $x \in L_{\text{out}}$, compute $G \otimes \phi(x)$, if $G \otimes \phi(x) < 0$, $switchIn(x)$; If a pixel $x \in L_{\text{in}}$ satisfies all the neighboring pixels in $\phi$ are non-positive, delete it from $L_{\text{in}}$ and set $\phi(x) = -3$.
2. For each pixel $x \in L_{\text{in}}$, compute $G \otimes \phi(x)$. If $G \otimes \phi(x) > 0$, $switchOut(x)$; If a pixel $x \in L_{\text{out}}$ satisfies all the neighboring pixels in $\phi$ are non-negative, delete it from $L_{\text{out}}$ and set $\phi(x) = 3$.
3. Check the stopping condition; if it is not satisfied, continue Cycle 2.

Algorithm 2. Object tracking stage of the FTCNB method.

Step 1: Initialization
1. Set the parameters: $n_{c}$, $N_{g}$, $\omega_{c}$ and $\omega_{g}$;
2. Perform superpixel segmentation for all the frames;
3. Generate the target-background saliency map for the first frame. Initialize $L_{\text{in}}$, $L_{\text{out}}$ and $\phi$ in the first frame according to the saliency map;
4. Generate the feature pools based on superpixels for the first frame: $C_{t}^f$, $C_{t}^{b}$, $T_{f}^{t}$ and $T_{b}^{t}$.

Step 2: Object contour tracking

for $t = 2$ to the end of the sequence do
1. $F_{t}^{d}$ is defined by Eq. (6);
2. Fast two-cycle level set curve evolution (Algorithm 1);
3. Save $L_{\text{in}}$, $L_{\text{out}}$ and $\phi$ for further refinement;
4. Update the color and texture feature pools at time $t$: $C_{t}^f$, $C_{t}^{b}$, $T_{f}^{t}$ and $T_{b}^{t}$.
end for

Algorithm 1 details the process of level set curve evolution, which consisting of two cycles. The first cycle is for boundary curve evolution. The second cycle is for boundary curve smoothness, where a $N_{g} \times N_{g}$ Gaussian filter $G$ is utilized. The stopping criterion is either of the following: (a) all the pixels in $L_{\text{out}}$ have larger probabilities belonging to the background and all the pixels in $L_{\text{in}}$ have larger probabilities of belonging to the foreground. (b) The maximum iteration number $n_{\text{it}}$ is reached.

Algorithm 2 describes the object tracking stage of the FTCNB level set method, which consists of two steps. In the first step, we

---

Fig. 4. Curve evolution process between adjacent frames.
first perform the parameter settings for: maximum iteration number \( m \), for curve evolution, Gaussian filter size \( N_g \), width of close background \( \omega_c \) and width of ambiguous pixels \( \omega_A \), we perform the superpixel segmentation for each frame as pre-processing. The target-background saliency map is then generated for the first frame based on superpixels. The foreground contours \( (L_{in} \) and \( L_{out} \)) and kernel matrix \( (\phi) \) are initialized according to the confidence map. The superpixel-based feature pools are also initialized: \( C^j, C_b^j, T^j \) and \( T^b \). In the second step, the speed function \( \mathbf{F}_j \) is computed according to Eq. (6). The curve evolution algorithm is then implemented to get the resulted \( L_{in}, L_{out} \) and \( \phi \), which are saved for further contour refinement. The feature pools are updated for each iteration. An example of curve evolution process between adjacent frames is depicted in Fig. 4.

3. Contour refinement stage of FTCNB method

The object tracking stage of the FTCNB method described in Section 2 is effective in most conditions. However, the feature pools of that stage are based on superpixels or K-means clustering of edge histogram descriptors. If these mid-level features are similar inside and outside the foreground contours, it may fail to get accurate tracking results. In this section, we introduce the contour refinement stage of FTCNB method for more accurate tracking. The new speed function is defined by the ratio of likelihood models of MAP expressions. The optical flow map between adjacent frames is utilized as temporal assistance.

3.1. Energy function definition

In the contour refinement stage, we apply the MAP method introduced by Mansouri [15,28] to define the energy function \( E_2 \). The MAP method uses the Bayesian theorem and the image pixels are assumed to be conditional independent:

\[
\Omega^*_{in} = \arg\max \Omega_{in} p\left( \Omega_{in} \mid I_{t-1}, I_t, \Omega_{in}^{t-1} \right) = \arg\max \Omega_{in} \prod_{x \in \Omega} p\left( I(x) \mid I_{t-1}, \Omega_{in}^{t-1}, \Omega_b^{t-1} \right) p\left( \Omega_{in} \mid I_{t-1}, \Omega_{in}^{t-1} \right),
\]

where \( I_t \) and \( I_{t-1} \) denote the current and previous frames, respectively. \( \Omega_{in}^{t-1} \) and \( \Omega_{in}^{t-1} \) are the \( m \) foreground regions of current and previous frames respectively. \( D \) is the image domain. Probability \( p(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_{in}^{t-1}) \) is the likelihood of observing a particular color at space-time location \((x, t)\). The likelihood model is partitioned into three parts according to the location of \( x \) in the image domain. The likelihood model of the foreground is defined by \( p_c(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_{in}^{t-1}) \). The likelihood models of the close and far background regions are defined by \( p_f(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_{in}^{t-1}) \) and \( p_f(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_{in}^{t-1}) \), respectively. Prior probability \( p(\Omega_{in} | I_{t-1}, \Omega_{in}^{t-1}) \) possesses prior knowledge regarding the foreground shape.

Based on these definitions, a new energy function at time \( t \) can be defined as follows:

\[
E_2 = -\int_{\Omega_{in}} \log p_c(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_b^{t-1}) \, dx
- \int_{\Omega_{in}} \log p_f(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_{in}^{t-1}) \, dx
- \int_{\Omega_{in}} \log p_f(I_t(x) | I_{t-1}, \Omega_{in}^{t-1}, \Omega_b^{t-1}) \, dx
- \log p(\Omega_{in} | I_{t-1}, \Omega_{in}^{t-1})
\]

(10)

where \( \Omega_{in}^{t-1} \) and \( \Omega_{in}^{t-1} \) are the \( m \) foreground and close background regions at time \( t \), \( \Omega_b^{t} \) is the far background at time \( t \).

3.2. Speed function definition

The energy function \( E_2 \) is not designed for minimization but for the deduction of the new speed function. In the object tracking stage of FTCNB method, the speed function is defined based on mid-level vision cues, i.e. superpixels and K-means clustering of edge histogram descriptors. To get more accurate results, we introduce the new speed function based on single pixels in the contour refinement stage.

![Fig. 5. Optical flow maps generation.](image-url)
Concerning the energy function $E_p$, we consider the prior model and the likelihood model of far background as already optimized, for the coarse contours are obtained through the object tracking stage. So the new speed function can be defined by the logarithmic ratio of the likelihood model of foreground to that of the close background:

$$F_d^p = \log \frac{p_{FG}(l(x)|l_{t-1}, \Omega_m^{-1}, \Omega_n)}{p_{BG}(l(x)|l_{t-1}, \Omega_m^{-1}, \Omega_n)}.$$  \hfill (11)

The optical flow map is applied as a temporal assistance to compute these likelihood models, based on which the correspondences between pixels of adjacent frames are constructed. In our work, the optical flow map is generated by Bruhn’s method [29]. Samples of optical flow maps are displayed in Fig. 5, in which color denotes the direction while brightness denotes the magnitude of optical flow. We assign variance value $\sigma_v^2$ for the optical flows. Moreover, the variance value $\sigma_v^2$ denoting the pixel color difference corresponded by the optical flows is also introduced.

Assuming the transformation from $l_{t-1}$ to $l_t$ can be denoted by the motion field $v$ and a Gaussian noise $b$:

$$l_t(x + v(x)) = l_{t-1}(x) + b(x),$$  \hfill (12)

s.t. $b \sim (0; \sigma_v^2)$, $v \sim N(\psi_0(x); \sigma_v^2)$.

where $\sigma_v^2$ and $\sigma_v^2$ are variances for the Gaussian noise $b$ and motion field $v$ respectively. $\psi_0(x)$ is the optical flow between $l_1$ and $l_{t-1}$ at position $x$. The likelihood models of foreground and close background are defined by Eqs. (13) and (14), respectively:

$$p_{FG}(l(x)|l_{t-1}, \Omega_m^{-1}, \Omega_n) \approx \sup_{y \in \Omega_m^{-1}} \left\{ \exp \left( - \frac{1}{2} \| l(x) - l_{t-1}(y) \|^2 (2 \sigma_v^2)^{-1} \right) \right\},$$ \hfill (13)

$$p_{BG}(l(x)|l_{t-1}, \Omega_m^{-1}, \Omega_n) \approx \sup_{y \in \Omega_m^{-1}} \left\{ \exp \left( - \frac{1}{2} \| | l(x) - l_{t-1}(y) \|^2 (2 \sigma_v^2)^{-1} \right) \right\},$$ \hfill (14)

3.3. Curve evolution of the contour refinement stage

The contour refinement stage of the FTCNB method is described in Algorithm 3. In step 1 of the algorithm, we set the parameters of $n_M$, $N_f$, $a_M$, $a_C$ and $\omega_t$ for initialization. In step 2, we first load the results of object tracking stage (Algorithm 2) for each frame. The optical flow map from current frame to the previous frame is then generated. Subsequently, the fast two-cycle curve evolution process is implemented with the definition of new speed function $F_d^p$.

The optical flow map is employed to correspond adjacent frames at single pixel level. It is complementary to the object tracking stage which is based on the mid-level features, e.g. superpixels. With the help of the optical flow map, more accurate foreground contours can be obtained through the contour refinement stage.

Algorithm 3. Contour refinement stage of the FTCNB method.

**Step 1: Initialization**
Set the parameters: $n_M$, $N_f$, $a_M$ and $\omega_t$.

**Step 2: Object contour refinement**
for $r=2$ to the end of the sequence do
1. Load $L_{iso}$, $L_{scale}$, $\phi$ from Algorithm 2;
2. Compute the optical flow map from $I_i$ to $I_{t-1}$. Get $\psi_0(x)$ at each pixel location $x$;
3. $F_d$ is defined by Eq. (11);
4. Fast two-cycle level set curve evolution (Algorithm 1). end for

4. Experiments

Our experiments are implemented using MATLAB R14 on a personal computer with 2.0 GHz Processor and 2.0 GB RAM. We first analyze the impact of the parameter setting. The comparisons of the proposed algorithms are then implemented, followed by the qualitative and quantitative comparisons with the baseline methods.

Since the background of our work is to track the vehicles from road image sequences, the datasets we selected are mainly focused on vehicle images. The MIT-CSAIL Car dataset [30] is employed for this purpose. The I AIR Traffic Road dataset is also chosen for experiments, which is constructed by the Institute of Artificial Intelligence and Robotics, Xi’an Jiaotong University. As a supplement, we also design experiments on the non-vehicle datasets, e.g. SegTrack dataset [31].

4.1. Parameter settings

The parameter settings are crucially important to our algorithms. The specification of parameters is considered as a training process before we analyze the accuracy and the speed of our methods. These parameters include $n_M$ for the maximum iteration number of curve evolution algorithm, $N_f$ for the Gaussian filter size, $a_M$ for the ambiguous pixel width, and $a_C$ for the close background width. $n_M$, $N_f$ and $a_M$ influence Algorithm 1. $a_C$ takes effects for Algorithms 2 and 3, i.e. the definitions of speed functions. These parameters should be set properly before the execution of the proposed algorithms.

To evaluate these parameters, we design experiments based on three image sequences from I AIR Traffic Road dataset: Black Car 1 ($512 \times 512$ pixel size, with 100 frames), Red Truck ($512 \times 512$ pixel size, with 200 frames) and Two Cars 1 ($512 \times 512$ pixel size, with 50 frames), as well as MIT-CSAIL Car dataset ($640 \times 480$ pixel size, with 50 frames). Although we generate the contour results for both the stages of object tracking and contour refinement, we mainly assess the results of contour refinement stage for the parameter settings.

During the evaluation, we apply the average tracking error (ATE) as the metric. For each pixel on the resulted contour of $\psi_0$ frame, we find its nearest pixel on the ground truth contour. The absolute distance between them is accumulated as tracking error (TE). The total tracking error (TTE) for an entire sequences with $N$ frames is defined as $TTE = \sum_{i=1}^{N} TE_i$. The ATE of an image sequence is then computed by: $ATE = TTE/N$.

4.1.1. Maximum iteration number

The maximum iteration number $n_M$ is used as one of the stopping conditions for the fast two-cycle curve evolution algorithm. To implement the algorithms on the above datasets, $n_M$
can be set in the domain of [10,30]. After extensive experiments and comparisons, we find that the ATE values decrease monotonically with the increase of \( n_M \). However, the ATE values decrease little if \( n_M \) is set larger than 20. For the speed concern, we choose \( n_M = 20 \) for the above datasets, which is effective in most conditions.

### 4.1.2. Width of close background

The parameter of the close background width \( \omega_C \) determines the perception of the background features and the definition of speed function, which is typically set to 10, 20 and 30. We analyze the impact of \( \omega_C \) with different combinations of Gaussian filter size \( N_g \times N_g \). The parameter \( N_g \) can be set to 3, 4 and 5. The evaluation of the combinations of \( \omega_C \) with \( N_g \) is shown in Fig. 6. Before the experiments, the parameter \( n_M \) is set to 20. The parameter \( \omega_A \) is set randomly, e.g. \( \omega_A = 4 \). The experiments are based on the above four image sequences. For each sequence, we apply the ATE value as the metric. Fig. 6(a)–(c) corresponds to different Gaussian filter sizes (3–5). As the experimental results show, the least ATE can be reached for all the sequences if \( \omega_C \) is set to 20. The optimum parameters for the four sequences are as follows: MIT-CSAIL (ATE = 11.0, \( \omega_C = 20 \), \( N_g = 4 \)), Black Car I (ATE = 6.9, \( \omega_C = 20 \), \( N_g = 3 \)), White Truck (ATE = 11.6, \( \omega_C = 20 \), \( N_g = 4 \)) and Two Cars I (ATE = 12.1, \( \omega_C = 20 \), \( N_g = 4 \)).

#### 4.1.3. Width of ambiguous pixels

After the specification of the parameters \( n_M \), \( \omega_C \) and \( N_g \), we then evaluate the impact of the width of ambiguous pixels \( \omega_A \). In Fig. 7, the ATE values correspond to different \( \omega_A \) are compared. \( \omega_A = 0 \) means the ambiguous pixels are omitted. As the comparison results demonstrate, the minimum ATE can be reached with \( \omega_A = 3 \) for the sequences of Black Car I and Two Cars I. For the sequences of MIT-CSAIL and White Truck, the optimum parameter is \( \omega_A = 4 \). Since the iteration speed of the algorithms slows down significantly with the increase of \( \omega_A \) we set \( \omega_A = 3 \) for each sequence for the concern of both accuracy and speed.

The above parameters are effective for the MIT-CSAIL dataset and the IAIR Traffic Road datasets. These parameters are set before we analyze the accuracy and the speed of our algorithms. For other datasets, e.g. SegTrack dataset, the parameters can be specified in a similar way.

### 4.2. Analysis of the proposed algorithms

After the specification of the parameters, we then analyze the experimental results of the proposed algorithms. The datasets utilized in Section 4.1 are chosen for the experiments: Black Car I, Red Truck, Two Cars I and MIT-CSAIL Car dataset. For our methods, the results of object tracking stage are denoted by Ours(T), while those of the contour refinement stage are denoted by Ours(T+R).

Fig. 8 depicts the results of Ours(T). The blue and red lines denote the two linked lists of contour pixels: \( L_{out} \) and \( L_{in} \), respectively. As illustrated in the third row, the resulted contours may split into multiple parts according to the deformable shapes. This is the characteristic of level set method.

In the left-column of Fig. 9, the tracking results of Ours(T) and Ours(T+R) are compared. The red line denotes \( L_{out} \) of Ours (T) while the green line denotes \( L_{in} \) of Ours(T+R). The comparison results prove that more accurate curves can be obtained if we apply the contour refinement stage. The segmentation results can be easily achieved based on the tracked contours. The
regions inside and outside the contours are treated as foreground and background, respectively. The segmentation results based on the contour refinement stage are depicted in the right-hand column of Fig. 9, where the green masks are covered on the background.

4.3. Comparisons with baseline methods

After the analysis of the proposed algorithms, we perform the qualitative and quantitative comparisons with the baseline tracking methods which include Scribble Tracker [4], SeamSeg [7], Video SnapCut [6] and Shi’s method [21]. The experiments are based on MIT-CSAIL dataset and three sequences from the IAIR Traffic Road dataset: Black Car II (512 × 512 pixel size, with 100 frames), White Truck (512 × 512 pixel size, with 200 frames) and Two Cars II (512 × 512 pixel size, with 50 frames). For the proposed method, we depict the inner contours \( L_i \) of both the object tracking stage and the contour refinement stage, denoted by Ours(T) and Ours(T+R) respectively. Shi’s method is implemented based on the transformed gray-scale image sequences. The shadows of the cars are considered as the background in the ground-truth.

The qualitative comparison results based on the MIT-CSAIL dataset and the Black Car II sequence are shown in Figs. 10 and 11, respectively. The comparison results demonstrate that the object tracking stage of our method can get accurate results in general. Further with the contour refinement stage, our method gets the most accurate results over all the baseline methods.

The comparison curves with the baseline methods are plotted in Fig. 12, with a sampling step of 5 frames. We apply the tracking error \( (TE) \) of each frame as the metric, where the ground truth contours are annotated in advance. For each pixel on the resulted contour of \( i \) frame, we find its nearest pixel on the ground truth contour. The absolute distance between them is accumulated as tracking error \( (TE) \). The comparison curves demonstrate that the object tracking stage of our method (Ours(T)) gets approximately similar results to the SeamSeg method. With the contour refinement stage, our method (Ours(T+R)) is more accurate and stable than all the other baseline methods. In Fig. 12(a), the tracking of the Scribble Tracker is lost at Frame number 23. Shi’s tracking result is lost at Frame number 37. The tracking result of Fig. 12(d) is lost at Frame number 22 for Scribble Tracker and at Frame number 83 for Shi’s method.

Furthermore, we quantitatively evaluate the segmentation

Fig. 8. Object tracking stage of the FTCNB method. The blue and red lines denote the two linked lists of contour pixels: \( L_{out} \) and \( L_{in} \). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)
results. Based on the tracking results of each frame, the region inside the contour is considered as foreground, while the region outside is the background. The error measure shown in Table 1, is the average number of pixels mis-labelled per frame. The error is defined as \( e(S) = \frac{\text{DORBGCE} \Delta S}{GT} \) [32], where \( S \) is the segmentation output, \( GT \) is the ground-truth segmentation and \( F \) is the total number of frames. The bold values in the table correspond to the best results obtained with the given datasets. In the last row of Table 1, the mean values of all the methods are compared. The proposed FTCNB method with contour refinement (Ours(T + R)) outperforms the baseline methods in all the datasets as well as giving the lowest overall error. This proves that our method is more robust in getting efficient segmentation results.

In Table 2, we compare the average tracking time of our methods with the baseline methods. As the results demonstrate, the speed of the contour refinement stage of our method (Ours(T)) is faster than all the other methods. The time cost plus the contour refinement stage (Ours(T + R)) is higher, approximately similar to Scribble Tracker. The reason is that the computation of the optical flow maps is time consuming. Although the tracking accuracies of SnapCut and SeamSeg are relatively higher, their tracking speed is much slower than ours.

The overall comparisons concerning both the accuracy and speed prove that Ours(T) is accurate in most conditions with the fastest speed. Ours(T + R) reaches the most accurate results, however, it is relatively more time consuming. For practical applications using our methods, a choice can be made between ours (T) and ours(T + R) with respect to speed and accuracy.

4.4. Experiments on SegTrack dataset

Aside from the vehicle datasets, we also perform experiments based on SegTrack dataset [31] as a supplement. Four sequences
from this dataset are chosen: Cheetah (320 × 240 pixel size, with 29 frames), Girl (400 × 320 pixel size, with 21 frames), MonkeyDog (320 × 240 pixel size, with 71 frames) and Parachute (414 × 352 pixel size, with 51 frames). The qualitative results of the contour refinement stage of our method (Our(T+R)) are shown in Fig. 13. For quantitative evaluation, we also apply the average number of pixels mis-labelled per frame as the metric [32]. According to the tracking results, the regions inside and outside the contour are considered as foreground and background, respectively. The comparisons with the methods of Scribble Tracker [4], Video SnapCut [6], SeamSeg [7] and Shi’s method [21] are shown in Table 3, which prove that our methods get the best segmentation results over the above baseline methods.
Fig. 11. Experiments based on the Black Car II sequence. In the ground-truth, shadows are considered as the background regions. (a) Ours(T). (b) Ours(T + R). (c) Scribble Tracker. (d) SnapCut. (e) SeamSeg. (f) Shi’s.
Fig. 12. Quantitative comparisons with the baseline methods. (a) MIT-CSAIL Car Sequence. (b) Black Car II Sequence. (c) White Truck Sequence. (d) Two Cars II Sequence.

Table 1
Comparison of average number of error pixels per frame.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MIT CSAIL</td>
<td>192</td>
<td>178</td>
<td>662</td>
<td>282</td>
<td>215</td>
<td>664</td>
</tr>
<tr>
<td>Black Car</td>
<td>224</td>
<td>188</td>
<td>466</td>
<td>202</td>
<td>258</td>
<td>565</td>
</tr>
<tr>
<td>White Truck</td>
<td>232</td>
<td>168</td>
<td>683</td>
<td>269</td>
<td>262</td>
<td>642</td>
</tr>
<tr>
<td>Two Cars II</td>
<td>226</td>
<td>154</td>
<td>483</td>
<td>467</td>
<td>217</td>
<td>477</td>
</tr>
<tr>
<td>Mean</td>
<td>218.5</td>
<td>172.0</td>
<td>573.0</td>
<td>305.0</td>
<td>238.0</td>
<td>587.0</td>
</tr>
</tbody>
</table>

Table 2
Average time for tracking each frame.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MIT CSAIL</td>
<td>0.2250</td>
<td>0.5257</td>
<td>0.5311</td>
<td>2.1250</td>
<td>1.1200</td>
<td>0.3284</td>
</tr>
<tr>
<td>Black Car</td>
<td>0.2037</td>
<td>0.5702</td>
<td>0.5972</td>
<td>2.4780</td>
<td>1.5700</td>
<td>0.3577</td>
</tr>
<tr>
<td>White Truck</td>
<td>0.2062</td>
<td>0.5734</td>
<td>0.6245</td>
<td>2.5980</td>
<td>1.7100</td>
<td>0.4059</td>
</tr>
<tr>
<td>Two Cars II</td>
<td>0.2521</td>
<td>0.6733</td>
<td>0.8629</td>
<td>2.5570</td>
<td>1.4800</td>
<td>0.3688</td>
</tr>
</tbody>
</table>
Furthermore, we conduct the comparison with the methods of Zhang et al. [5], Varas et al. [8], Tsai et al. [32] and Ma et al. [33], as shown in Table 4. The results shown are taken from the respective author’s publications. The proposed method with contour refinement gets the most accurate segmentation results in three out of four sequences. In the Cheetah sequence, our method performs less accurate than the method of Varas et al. The reason is that the skin of the animal is similar to the ground in color and texture information. Moreover, the poor superpixel segmentations on the animal legs lead to this result.

5. Conclusion and future work

The contour tracking problem has been studied extensively by researchers in the communities of signal processing and computer vision. The level set method is a powerful tool for curve evolution and fitting of the foregrounds. The proposed method possesses the attributes of the typical level set method, e.g., adjusting to deformable shapes even when the foreground separates into multiple parts. However, our method is faster than the traditional level set methods for the PDEs need not be solved.

In our work, the speed function for the curve evolution is
defined by the region competition terms on contour pixels rather than by solving PDEs. The speed function is first computed according to superpixel color distributions and the correlation coefficients of texture histograms. Once we apply define MAP expressions with the assistance of optical flow maps, a new speed function is calculated between the likelihood models. Motivated by the speed functions, the fast two-cycle curve evolution process is implemented, which consists of one cycle for contour pixel movement and a second cycle for contour smoothness.

Our approach depends on the feature differences between the foreground and the close background regions. Thus, this approach may fail under conditions in which the two regions are very close in terms of color and texture information. Moreover, tracking errors may accumulate for the feature pools are propagated from the previous to the current frames.

Directions for the improvement of our approach are as follows:

- **More definitions and combinations of the feature descriptors:** In this study, the mid-level feature descriptors mainly include the superpixels, K-means clustering in CIE-LAB color space and the edge histogram descriptors. Future work can define additional features. For example, the texture feature can be defined by the Gabor filter [34]. Additionally, shape priors [35–37] can be considered as well.

- **Common feature pools:** Our approach can natural be extended to define common feature pools. Errors may be accumulated and propagated in the current framework, so common feature pools can be constructed for all the frames. This improvement can be applied to the initialization step of the object tracking stage.

- **Integration of global optimization:** In the future, the global optimization process can be utilized. The MAP method can be applied independently based on the greedy algorithm rather than on the region competition terms.

**Acknowledgment**

This work was supported in part by the National Natural Science Foundation of China under Grant nos. 61328303 and 61305051. Yaochen Li gives warm thanks to Prof. Stanley Osher and the Mathematics Department of the University of California, Los Angeles (UCLA) for offering the visiting opportunity from December 2012 to January 2014. The authors would like to thank Yanan Yu, Shiyu Cao, Dengkui Tian, as well as Dr. Kuniaki Kawabata for their helpful discussions and assistance. The authors thank the anonymous reviewers for their valuable comments and suggestions.

**References**


[31] (http://cpl.cc.gatech.edu/projects/SegTrack/).


