Noname manuscript No.
(will be inserted by the editor)

Motion Coherent Tracking Using Multi-label MRF Optimization

David Tsai · Matthew Flagg · Atsushi Nakazawa · James M. Rehg

Received: date / Accepted: date

Abstract We present a novel off-line algorithm for target segmentation and tracking in video. In our approach, video data is represented by a multi-label Markov Random Field model, and segmentation is accomplished by finding the minimum energy label assignment. We propose a novel energy formulation which incorporates both segmentation and motion estimation in a single framework. Our energy functions enforce motion coherence both within and across frames. We utilize state-of-the-art methods to efficiently optimize over a large number of discrete labels. In addition, we introduce a new ground-truth dataset, called Georgia Tech Segmentation and Tracking Dataset (GT-SegTrack), for the evaluation of segmentation accuracy in video tracking. We compare our method with several recent on-line tracking algorithms and provide quantitative and qualitative performance comparisons.

Keywords Video Object Segmentation · Visual Tracking · Markov Random Field · Motion Coherence · Combinatoric Optimization · Biotracking

The authors are affiliated with the Center for Behavior Imaging and the Computational Perception Laboratory within the School of Interactive Computing at the Georgia Institute of Technology. The third author was a Visiting Scientist in the School of Interactive Computing during the time this research was conducted. He is currently affiliated with the Cybermedia Center, Osaka University.

David Tsai
E-mail: caihsiaoster@gatech.edu

Matthew Flagg
E-mail: mflagg@gmail.com

Atsushi Nakazawa
E-mail: nakazawa@cmc.osaka-u.ac.jp

James M.Rehg
E-mail: rehg@cc.gatech.edu

1 Introduction

Recent work in visual target tracking has explored the interplay between state estimation and target segmentation [2,15,35]. In the case of active contour trackers and level set methods, for example, the state model of an evolving contour corresponds to a segmentation of target pixels in each frame. One key distinction, however, between tracking and segmentation is that tracking systems are designed to operate automatically once the target has been identified, while systems for video object segmentation [46,44,10] are usually interactive, and incorporate guidance from the user throughout the analysis process. A second distinction is that tracking systems are often designed for on-line, real-time use, while segmentation systems can work off-line and operate at interactive speeds.

Several recent works have demonstrated excellent results for on-line tracking in real-time [2,15]. However, the quality of the segmentations produced by on-line trackers is in general not competitive with those produced by systems for interactive segmentation [46,33,31], even in cases where the user intervention is limited. One reason is that segmentation-based methods often adopt a global optimization method (e.g. graphcut) and explicitly search a large, fine-grained space of potential segmentations. In contrast, for tracking-based methods the space of possible segmentations is usually defined implicitly via the parameterization of the target model, and segmentation accuracy may be traded for computational efficiency.

Our work is motivated by applications in biotracking [32,25,1,42,3,8,16, 17], where there is a need for a general purpose tool for tracking a wide range of animals with different morphologies. In these applications, an off-line batch formulation of video analysis is acceptable, but the need for guidance by the user must be minimized in order for the tool to be useful to biologists. Standard on-line tracking methods which use a coarse model of the target shape and focus on robust estimation of the target center of mass are not sufficient for many biotracking applications, which require a more detailed segmentation in order to automate the measurement of experimentally-relevant behaviors. However, while it is important, for example, to be able to extract the limbs of a target animal, biotracking applications do not require the pixel-accurate segmentations that are needed in video post-production and special effects domains.

This paper describes a new method for automatic target segmentation and tracking which uses a multi-label Markov Random Field (MRF) formulation to sequentially "carve" a target of interest out of a video volume. Our goal is to obtain higher-quality segmentations than existing on-line methods, without requiring significant user interaction. The primary novelty of our approach is our treatment of the inter-related tasks of segmenting the target and estimating its motion as a single global multi-label assignment problem. Energy functions enforce the *temporal coherence* of the solution, both spatially and across time. The result is a clean problem formulation based on global energy minimization. In contrast, on-line tracking methods often employ a diverse set

of techniques to achieve good performance, including adaptive cue combination [2], spatially-varying appearance models [15], and shape priors [11]. We demonstrate experimentally that our approach can yield higher-quality segmentations than these previous methods, at the cost of greater computational requirements within a batch formulation.

A second goal of this work is to support the quantitative assessment of segmentation quality in tracking, through the development of a standardized database of videos with ground-truth segmentations. There has been no systematic quantitative or comparative evaluation of segmentation quality within the visual tracking literature.[1] We identify three properties of video sequences that can hamper segmentation: color overlap between target and background appearance, interframe motion, and change in target shape. We have developed a quantitative measure for each of these properties, and have assembled an evaluation dataset, called *GT-SegTrack*, which spans the space defined by these challenges. We provide a quantitative and qualitative evaluation of our method and compare it to two recent on-line contour-based trackers [2, 15] and two reference systems for video segmentation using graphcuts.

In summary, this paper makes four contributions:

- We introduce a novel multi-label MRF formulation of video tracking which provides high-quality target segmentations.
- We propose an energy function that enforces motion coherence between spatial neighbors and across the temporal dimension.
- We present a novel database that supports systematic quantification of segmentation quality with respect to three types of challenges found in real-world video footage.
- We describe a comprehensive set of performance comparisons to four existing methods for video segmentation and tracking.

2 Related Work

There are two bodies of previous work which are related to our method. The first are techniques for video object segmentation and layer segmentation which also make use of an MRF formulation. The second are tracking methods which construct a segmentation of the target object.

Video object segmentation has been previously formulated as a binary labeling problem in an MRF and solved using a volume graphcut.[2] A key issue in this approach is the construction of the temporal links that connect pixel sites across frames. In the classical formulation from [7], the MRF is instantiated in the temporal dimension by linking corresponding pixel sites in adjacent frames. An alternative formulation was proposed by Li et. al. [31], in which temporal links were instantiated between superpixels in adjacent

[1] In contrast, there has been extensive work on comparing the state estimation performance of standard state-based trackers. Some representative examples are [24] and the VS-PETS workshops [14].

[2] Volume graphcuts have also been employed in medical image segmentation [5,30].

frames. These alternatives and others are illustrated in Figure 4. In section 5.1, we present an experimental comparison between these competing alternatives that provides a baseline for evaluating our proposed method. A key difference between our formulation and these earlier works is our incorporation of motion and segmentation constraints into a single, unified multi-label formulation.

Several alternative formulations of temporal coherence have been employed in video segmentation, such as KLT Tracking [47,33], SIFT matching [46] and dense optical flow [23]. These methods depend heavily on the quality of the motion estimates and may fail in challenging sequences. Furthermore, the flow in these works is primarily calculated between pairs of frames, and does not exploit coherence over larger time windows. Other works which address the joint computation of optical flow and segmentation [49,13] are based on iterative estimation methods which do not provide any global guarantees on solution quality. Other works which combine tracking and segmentation include [35, 18,48].

Recently, there have been significant advances in discrete optimization methods for large label spaces [26,27]. Komodakis et. al. proposed a discrete optimization algorithm called Fast-PD [27], which provides an efficient approach to minimizing the discrete MRF energy. It has been used in image registration [21], stereo disparity estimation [27], and optical flow estimation [22]. In these latter applications, it is sufficient to analyze pairs of frames, while our case requires the analysis of the entire video volume. The problem of globally-optimal shape-based tracking has also been addressed recently using combinatoric optimization methods [19,37]. In biotracking applications, however, color is often an important cue, and the substantial variations in target shape over time can pose problems for purely shape-based methods.

A large number of on-line tracking methods can produce object segmentations (representative examples are [15,2,43]). Since these methods are fully-automatic, they represent an interesting point of comparison, even though they are not necessarily designed to produce fine-grained target segmentations. Bibby and Reid describe an impressive tracking system in [2], which demonstrates adaptation of the target model and integration of multiple cues so as to track a wide range of challenging targets. A level-set based system, described in [15], uses a combination of spatially-constrained appearance modeling and motion estimation to track the target boundary. In comparison to these works, we employ a volumetric multi-label MRF formulation. We provide experimental comparisons to these two methods in sections 5.2 and 5.3.

There has been a significant amount of prior work on tracking specific types of targets, such as people and animals, using kinematic models in conjunction with Bayesian filtering techniques. Representative examples include [12,40, 34,45,39,9]. While these works are capable of describing articulated animal movement in significant detail, the need to specify kinematic and dynamic models for target animals is a barrier to the widespread use of these methods by biologists. In contrast, segmentation-based methods have the advantage that the experimenter need only identify the target animal via manual segmentation in the first frame in order to begin tracking.

A preliminary version of the work described in this paper was presented in [41]. In comparison to that prior publication, this work includes a more complete experimental evaluation of the method and an expanded description of the method (see Algorithm 1 and 2).

3 Multi-Label MRF Framework

Given a video sequence and a segmentation of a target of interest in the first frame, our goal is to carve the moving target out of the video volume, yielding a target segmentation in every frame. We adopt the volumetric MRF formulation, in which the video volume is represented as a multi-label MRF with hidden nodes corresponding to the unknown labels. The resulting optimization problem is to find the joint label assignment L for all pixel sites G in the video volume that minimizes

$$\begin{aligned} E(L) = \lambda_1 & \sum_{p \in G} V_p(l_p) \\ + \lambda_2 & \sum_{p \in G} \sum_{q \in N_s(p)} V_{pq}(l_p, l_q) \\ + & \sum_{\substack{p_t \in G \\ q_\tau = N_t(p_t, l_{p_t})}} V_{p_t q_\tau}(l_{p_t}, l_{q_\tau}) \end{aligned} \tag{1}$$

where $L = \{l_p\}_{p \in G}$ is a global labeling, $V_p(\cdot)$ are the unary potentials representing the data term, $V_{pq}(\cdot, \cdot)$ are the pairwise potentials representing the smoothness term, G represents the set of pixel sites (nodes) in the video volume, λ_1 and λ_2 are tradeoff parameters, N_s represents the spatial neighborhood system of the nodes, and N_t identifies the temporal neighbor, which is determined by the assigned label. The definition of the temporal neighbor for a site p_t is given by

$$N_t(p_t, l_{p_t}) = p_t + D(l_{p_t}). \tag{2}$$

We now define the label space and energy terms used in Equation 1.

3.1 Definition of Label Sets

In contrast to the standard approach to MRF-based segmentation, our label set *augments* the usual foreground/background binary attribute with a discrete representation of the flow between frames. We define a quantized motion field $\{d^1, \cdots, d^i\}$ associated with each pixel (and ignoring boundary effects). Then the label assignment l_p to pixel site p has two components:

- An attribute $\text{Attr}(l_p) \in \{\text{fg}, \text{bg}\}$, which gives the segmentation label for p,
- a displacement $D(l_p) \in \{d^1, \cdots, d^i\}$ which gives the offset from p to the corresponding pixel in the next frame.

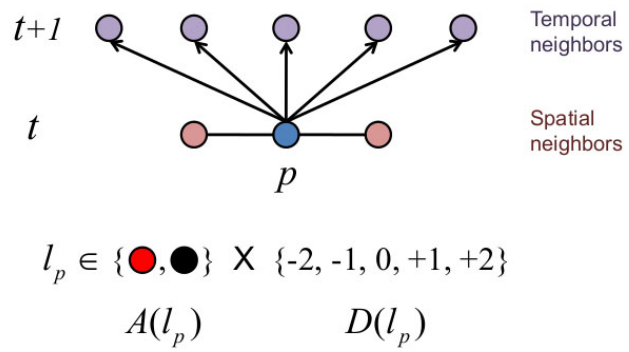

Fig. 1 Illustration of label definition. We illustrate the label space for a center pixel in frame t. If the maximum displacement in the x direction is 2, then there are 5 possible displacements ranging from (-2,0) to (2,0). In each case, the pixel can also be labeled either foreground (red) or background (black), resulting in 10 possible labels per pixel.

If the maximum possible spatial displacement in x or y is M, and all integer displacements are allowed, then there will be a total of $i = (2M + 1)^2$ flow possibilities for a single pixel in a 2D image (including zero displacement). In addition, each pixel can be either foreground or background, leading to a total of $2(2M+1)^2$ labels per pixel. Figure 1 illustrates these combinations for a simple 1D example. Note that we take the Cartesian product of attributes and flows (rather than their sum) because the interaction between these two components is a key element in enforcing temporal coherence between frames.

3.2 Data Term

The data term in Equation 1 is defined as follows:

$$V_p(l_p) = \underbrace{\int_{\Omega_p} w_p(x) \cdot \rho(I_t(x), I_{t+1}(x + D(l_p)))dx}_{\text{Appearance Similarity}} + \underbrace{U_p(l_p)}_{\text{Appearance Model}} \quad (3)$$

The first term in Equation 3 measures the appearance similarity across the temporal dimension. Ω_p represents the pixels in the local patch centered at p, $I(\cdot)$ is the intensity of the pixel, $w_p(x)$ is a weighting function centered at p, and $\rho(\cdot, \cdot)$ is the similarity measure. Our implementation uses the Gaussian weighted (with radius m_1) Sum of Absolute Differences between the two patches centered by control points. Other measures such as normalized cross correlation, Rank Filter, or mutual information, could also be used.

The second term measures pixel appearance relative to the foreground-background color models, and is defined as:

$$U_p(l_p) = \begin{cases} -\log \Pr(I(p)|\text{foreground}) & \text{Attr}(l_p) = \text{fg} \\ -\log \Pr(I(p)|\text{background}) & \text{Attr}(l_p) = \text{bg}. \end{cases} \quad (4)$$

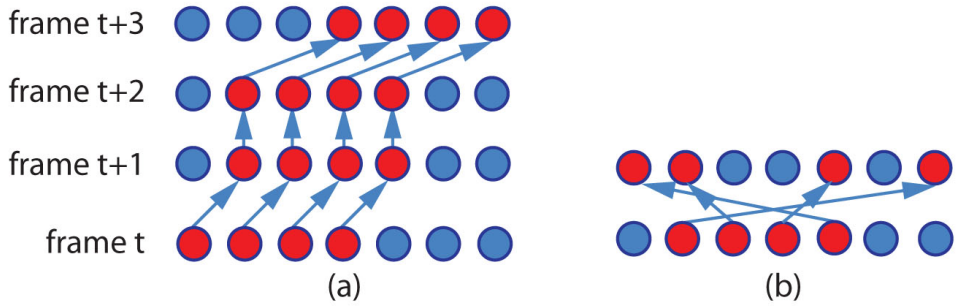

Fig. 2 Example of Motion Coherence. (a) The movement is both spatially and temporally coherent (b) The movement is spatially and temporally incoherent.

The appearance model term measures the likelihood of the pixel depending upon whether it is foreground or background. We employ a Gaussian mixture model with m_2 Gaussians for both foreground and background probability distributions in RGB color space. These models are used to compute the pixel likelihoods in Equation 4.

3.3 Smoothness Term

The smoothness term is the key part of our formulation. It incorporates coherence in attributes over time as well as spatial and temporal motion coherence. In Equation 1, the smoothness term is defined with respect to the spatial and temporal neighbors, $N_s(p)$ and $N_t(p)$, of each pixel p. An example of spatial and temporal neighbors is given in Figure 1. For each pixel site there are a total of 4 spatial neighbors and $(2M+1)^2$ possible temporal neighbors. Each neighbor contributes a binary potential function to the sum in Equation 1.

The basic coherence function is given in Equation 5. It is evaluated for each pair of spatial neighbors (p, q). In addition, it is evaluated for each pair of temporal neighbors $(p_t, N_t(p_t, l_{p_t}))$ (see Equation 1).

$$V_{pq}(l_p, l_q) = \underbrace{\lambda_3 |D(l_p) - D(l_q)|}_{\text{Motion Coherence}} + \underbrace{U_{pq}(l_p, l_q)}_{\text{Attribute Coherence}} \tag{5}$$

The first term of Equation 5 captures the property of motion coherence. In the case of spatial neighbors, the intuition behind this cost is that points which are close to one another will move coherently. In the case of temporal neighbors, the intuition is that the object should maintain a coherent movement across frames. This is illustrated in Figure 2.

The second term in Equation 5 captures the property of attribute coherence:

$$U_{pq}(l_p, l_q) = \begin{cases} E_c & \text{Attr}(l_p) \neq \text{Attr}(l_q) \\ 0 & \text{Attr}(l_p) = \text{Attr}(l_q). \end{cases} \tag{6}$$

When applied to temporal neighbors, this term models the interaction between segmentation and estimated motion, which is a key benefit of the joint label space illustrated in Figure 1. It penalizes labellings in which spatial and temporal neighbors receive different segmentations (i.e. pixel attributes). In the spatial domain, it enforces the constraint that adjacent pixels have the same attributes. This is identical to the spatial smoothness constraint used in the standard binary label MRF formulation. In the temporal domain, it enforces the constraint that the two pixels connected by a flow vector (i.e. temporal neighbors) should have the same attribute label.

3.4 Optimization

In order to optimize the energy function in Equation 1, we adopt the Fast-PD method of Komodakis et. al. [27,28].Fast-PD has demonstrated impressive performance in multi-label MRF optimization. The generated solution is guaranteed to be an f-approximation to the true optimum, and in practice the per-instance approximation factor often drops quickly to 1 [27]. Fast-PD utilizes alpha-expansions, and solves a max-flow problem for a series of graphs. Because it ensures that the number of augmentations per max-flow decreases over time, it can provide substantial speed-ups over conventional multi-label graphcut methods, which we have found to be unacceptably slow due to the size of our label space. In our experiments, we use the library described in [28].

Algorithm 1 Multi-Label Tracking Algorithm

```
start ← 0
while start + n < end do
   step 1: sample space-time window from frame start to start + n
   step 2: downsample pixels in every frame of the window by k
   step 3: establish hard constraints for frame start
   if start = 0 then
      initialized from human input
   else
      initialized from previous result on frame start
   end if
   step 4: set value for V_p(l_p) and V_pq(l_p, l_q)
   step 5: optimization with Fast-PD over a window of n + 1 frames
   step 6: use down-sampled segmentation result to set hard constrains and interpolate
   using graphcut
   step 7: start ← start + n
end while
```

In our implementation, we use a sliding window approach to address the practical infeasibility of storing the entire video volume graph in memory. Pseudocode for the proposed method is given in Algorithm 1. We first sample $n+1$ consecutive frames to form a space-time window (Step 1). For each sliding window position, the first frame of the current window is overlapped with the last frame of the previous window. This enforces the continuity of the solution

between window evaluations. The main consideration for the choice of n is not to exceed the available memory and computation time, which is part of the experiment design. Within each window, we spatially down-sample the image by a constant factor k to obtain a set of node control points (Step 2). By working with these control points first, and then up-sampling them to the original resolution in a post-processing step, we reduce the computational cost. Obviously, increasing k can significantly shorten running time, because the number of nodes and edges in the MRF will decrease dramatically. However, setting k too large will sacrifice accuracy, because too many details are missed as a result of the control points being too sparse. Hard constraints are then established for the first frame, using labels obtained from the previous volumetric label assignment (or from the initialization frame in the beginning) (Step 3). In this case, hard constraints mean that we assign a large constant penalty inf to the data term, to make sure the resulting segmentation is consistent with the initialization.

We then assign values to the data and smoothness terms, according to Equations 3 and 5 (Step 4). We build a Gaussian Mixture Model for the data term [5], and use the standard smoothness term as described in [6]. The trade-off is controlled by a parameter λ_2 on the data term. Next, we perform the global optimization to infer motion and segmentation variables simultaneously using Fast-PD (Step 5). We then use the standard graphcut to interpolate the down-sampled segmentation result to the original size (Step 6). In the interpolation, the hard constraints (similarly defined as in Step 3) are established using the binary segmentation results. We identify the largest connected component in the foreground segmentation, which presumably corresponds to the target of interest. We then refine the segmentation by removing the small blobs whose area is below a fraction p of the target area. Our implementation uses the following parameter values: $n = 4, k = 4, inf = 1e5, \lambda_1 = 0.2, \lambda_2 = 0.2, \lambda_3 = 1, E_c = 2, M = 16, m_1 = 5, m_2 = 5, p = 0.1$.

3.5 Baseline Method Using Explicit Temporal Links

To enforce motion coherence, we perform joint optimization of segmentation and motion estimation. A simpler alternative would be to estimate temporal correspondences directly and then use them to explicitly instantiate temporal edges in the graph. These temporal edges, in conjunction with spatial edges, would constitute a standard volumetric MRF in which hidden nodes correspond to the unknown labels $\{\text{fg}, \text{bg}\}$. The volumetric graph can then be segmented using the standard graphcut method for binary MRFs. The disadvantage of this approach is that motion estimates might be unreliable, leading to suboptimal graph structures and inaccurate segmentations.

In order to demonstrate the benefit of the joint approach over the decoupled paradigm, we implemented a baseline graphcut method for comparison purposes. Conceptually, there are two parts to the baseline algorithm:

1. Instantiation of the temporal n-links that define the volumetric graph.

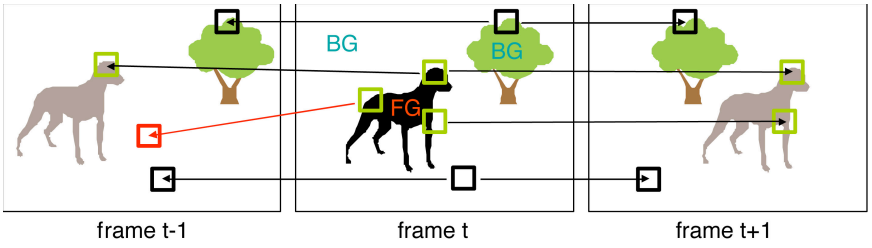

Fig. 3 Temporal coherence from tracked features. Nodes (pixels) in the central frame are connected to the other frames' nodes according to KLT flow. Temporal n-links are entered from a patch of pixels at frame t to its corresponding patch at $t^{'}$.

2. Execution of graphcut segmentation in a sliding temporal window, in order to satisfy memory constraints.

The energy function which is minimized in the baseline approach is similar to that of Equation 1, but with two important differences. The first difference is that we only consider a binary attribute label, in contrast to the joint label space used in Equation 1. The second difference lies in the method of instantiating temporal links. In the baseline approach, the temporal neighborhood $N_t(p_t)$ is specified before the optimization stage, and does not depend upon the label assignment as it does in Equation 2.

Figure 4 illustrates four different ways to instantiate the temporal links. The first three of these have appeared in previously-published works, while the dynamic temporal n-link method shown in Figure 4 (E) is a new approach which is described in this section. In section 5.1, we present an experimental evaluation of these four different approaches. We demonstrate that the dynamic approach (E) gives the best performance among the four. In section 5.2, this method, which we refer to as the volume graphcut approach, serves as a baseline for evaluating the joint label assignment approach.

The dynamic n-link approach uses feature detection and tracking to adaptively instantiate the temporal links, thereby adapting the volumetric graph to the motion in the video. This process is illustrated in Figure 3. The video volume is analyzed by means of a sliding window. For each window position, a standard feature detector [38] and KLT tracker [4] is used to identify correspondences between the central frame and all other frames in the local volume. Each identified correspondence between a pixel p in frame t and a corresponding site q in frame t' results in the creation of a temporal n-link in the volumetric graph. Note that outside of the differences in label space and link instantiation, the baseline cost function is the same as that in Equation 1. Specifically, the data and spatial smoothness terms are given by Equations 4 and 6, respectively.

Pseudocode for the baseline method is given in Algorithm 2. We first sample a total of $n+1$ consecutive frames centered at frame *middle* (Step 1). For each window of frames, we assign values for the data term $D_p(f_p)$ using Equation 4. To initialize the optimization, we set hard constraints using

either manually-labeled strokes on frame 0, or the labels from the previous segmentation if *middle* is not the first frame (Step 2). We assign values to the smoothness term $V_{pq}(f_p, f_q)$ using Equation 6. Standard 4-connected spatial n-links are entered for all nodes in the local graph (Step 3).

In order to enforce temporal coherence, we compute KLT flow from the middle frame to all other frames. To make a fair comparison to Algorithm 1 where the displacements are limited to a fixed range, we only retain the flows whose displacement in each direction is less than $d_{max} * k$, where k is the frame difference (Step 4). Costs associated with temporal n-links are specified by Equation 6 (Step 5). The energy function is optimized using the GraphCut algorithm (Step 6). After removing small blobs using the approach described in Algorithm 1, the segmentation of frame $middle + 1$ is output as the final result for that frame (Step 7), and the window is advanced by one frame. This continues until the end is reached (Step 8). In our implementation, we use $n = 4, inf = 1e5, \lambda_1 = 0.2, E_c = 2, d_{max} = 16, m_1 = 5$.

Algorithm 2 pseudo code of the baseline

$middle \leftarrow 0$
while $middle + n < end$ **do**
  step 1: sample space-time window from frame $middle - n/2$ to $middle + n/2$
  step 2: set value for $D_p(f_p)$ using Equation 4
  if $middle = 0$ **then**
    establish hard constrains for frame *middle* from human input
  else
    establish hard constrains for frame *middle* from previous result
  end if
  step 3: set value for $V_{pq}(f_p, f_q)$ in each frame using Equation 6
  step 4: compute KLT flow from frame *middle* to all other frames and retain those within maximum displacement.
  step 5: assign edge values to KLT flows for $V_{p_t q_r}(f_{p_t}, f_{q_r})$ using Equation 6, and remove outliers.
  step 6: optimization with GraphCut
  step 7: Output label for frame $middle + 1$
  step 8: $middle \leftarrow middle + 1$
end while

4 GT-SegTrack Database

An additional goal of this work is to facilitate a deeper understanding of the trade-offs and issues involved in on-line and off-line formulations of video segmentation and tracking. Since we were unable to identify any previous work addressing the comparative segmentation performance of tracking methods, we created our own standardized database of videos with ground-truth segmentations.[3] We began by identifying three properties of video sequences that pose challenges for segmentation:

[3] The dataset is available at http://cpl.cc.gatech.edu/projects/SegTrack

sequence	color	motion	shape	challenge summary
parachute	.038	.119	.024	low-low-low
girl	.205	.145	.147	low-low-high
monkeydog	.299	.243	.132	low-high-high
penguin	1.02	.016	.013	high-low-low
birdfall	.466	.283	.070	high-high-low
cheetah	.760	.273	.187	high-high-high

Table 1 GT-SegTrack database metrics: Associated with each sequence is a numerical measurement of its difficulty with respect to the challenges of color overlap, interframe motion and shape change. Each sequence is characterized as being either high or low difficulty with respect to the three challenges.

– *Color* overlap between target and background appearance,
– Interframe *motion*, and
– Change in target *shape* between frames.

We developed a quantitative measure for each of these phenomena, described below, and we systematically assembled an evaluation dataset, called *GT-SegTrack*, which spans the space of challenges. This dataset is used to quantify the performance of our method in section 5.2.

In order to obtain a set of sequences which adequately cover the space of challenge properties, we went through the following selection procedure. First, a set of 11 image sequences were manually identified as potentially covering the space of challenges. Then each sequence was manually rated as being either high or low with respect to each challenge type. The sequences were assigned to one of eight combination bins, corresponding to a rating of either high or low for each of the three challenges: color, motion, and shape.

Next, the sequences were manually segmented and the challenge measures were computed for each one. Finally, using the computed measures we selected six image sequences, ranging in length from 21 to 70 frames, that maximally cover the challenge space. Table 1 lists the resulting video clips and their numerical scores with respect to the challenge measures for color, motion, and shape. The last column summarizes the difficulty of the sequences with respect to the three challenges. For example, the *penguin* sequence is summarized as high-low-low, since it is challenging from the standpoint of color overlap, but presents a low level of difficulty with respect to interframe motion and shape change. We now describe the three challenge metrics.

Target-background color overlap: An accurate segmentation of the target, provided by the user, is commonly used to estimate a color model for the target and non-target pixels. However, the discriminative power of such models is inversely proportional to the degree of overlap between the figure-ground color distributions. Numerous trackers and interactive segmentation systems evaluate color overlap to decide how and when to lessen the importance of color and increase reliance on other models of the target (e.g. a locally modeled shape prior as in [46]). We chose to model target and ground colors with GMMs

containing 5 Gaussians. Equation 7 gives a formula for evaluating color overlap on a per-frame basis. High C_1 values correspond to large target-background overlap, which makes segmentation and tracking more difficult. The average measure per sequence is given in Table 1.

$$C_1 = \frac{\int_{X \in fg} p(X|bg)}{\int_{X \in fg} p(X|fg)} + \frac{\int_{X \in bg} p(X|fg)}{\int_{X \in bg} p(X|bg)} \tag{7}$$

Interframe target motion: Many tracking systems rely on the matching of discriminative local features to maintain temporal coherence. Large target motions result in an expanded search space for registration, which can result in poor matching performance. From ground truth segmentation, we measure interframe motion as the foreground XOR intersection area normalized by the mean object bounding box area. The per-frame average motion is reported in Table 1.

Target shape change: Shape priors constructed from target initialization, keyframes (as obtained automatically in [48]) and previously-segmented frames are often adaptively applied when other appearance models (e.g. color) are predicted to have small discriminative power. When target shape is relatively constant and motion estimation is reliable, shape priors can be used to track reliably in sequences with large figure-ground color overlap and occlusions [46]. However, when motion estimation is unreliable or shape change is drastic, this strategy can fail for obvious reasons. The GT-SegTrack database contains such challenging scenarios. The measurement of shape change is similar to that of target motion: it is given by the foreground XOR intersection area normalized by the mean object bounding box area after compensating for translational motion estimated from centroid differences. Table 1 reports the mean shape change for each sequence.

5 Experiments

We conducted four experiments. The first experiment evaluates the relative performance of 5 different temporal link instantiation strategies, as described in section 3.5. The second experiment provides quantitative comparisons between our method and three alternative approaches, using the GT-SegTrack Database. The three methods are: the level set-based tracker from [15], the Rotobrush tool from Adobe AfterEffects CS5, which seems to be based in part on the method of [46], and the baseline volumetric graphcut approach described in section 3.5 (see algorithm 2). In the third experiment, we provide qualitative performance comparisons to [2] and [15], demonstrating our method's ability to generate more accurate segmentations in many cases. The fourth experiment assesses our system's performance in tracking longer sequences.

In our experiments, each tracker was initialized by the ground-truth of first frame(if provided), or by performing a manual segmentation of the first frame into foreground and background pixels. We performed the same initialization

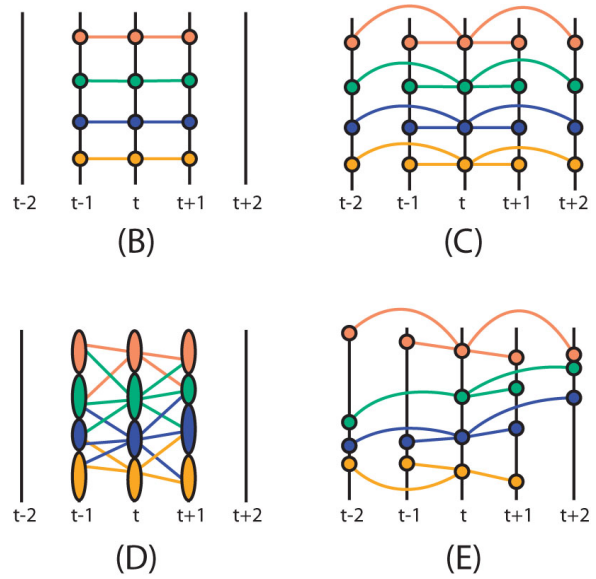

Fig. 4 Temporal n-link structures: (B) static temporal n-links in a 3 frame window, (C) static temporal n-links in a 5 frame window, (D) dynamic temporal n-links on super-pixel nodes as described in [31] and (E) dynamic temporal n-links instantiated from tracked features.

for our method, the baseline graphcut-based method, and the Adobe After-Effects Rotobrush tool. We initialized the level set algorithm by drawing a contour as described in [15]. In this case, the user specified an initial contour in the first frame that defined the target segmentation. For each competing method, we used a single set of manually-specified parameters in all of our experiments.

5.1 Temporal Link Instantiation

We investigated the performance of five different approaches, denoted as (A) through (E), for instantiating fixed temporal n-links. In comparison to the joint label approach, these graph structures are all specified prior to optimization during a pre-processing stage, and are not a function of the label assignment. We implemented and tested each method within our baseline framework of sweeps of local graphcuts. In approach (A), there are no temporal links, and so each frame is segmented independently. Approaches (B) through (E) are illustrated in Figure 4. The the most straight-forward case is (B), which was described in the classical work of Boykov and Jolly [7]. In this approach, temporally-adjacent sites are connected, resulting in temporal n-links of the form $((x,y,t),(x,y,t+1))$. One disadvantage of this formulation is that when the inter-frame motion is large, these fixed temporal links are less likely to

reflect true motion correspondences. Approach (C) is similar to (B), but increases the temporal interval over which links are instantiated.

An alternative to fixed temporal links is to adapt the link structure to the video contents. In the method of Li et. al. [31], super-pixels are computed in each frame, and links are constructed between temporally- and spatially-adjacent superpixels. This is shown as approach (D) in Figure 4. In particularly challenging sequences it may be useful to construct temporal links which more directly reflect the motion in the video. In particular, it may be useful to instantiate "long range" temporal links which cross multiple frames. This is accomplished by the method (E), which was presented in section 3.5.

Of the methods shown in Figure 4, two of them are novel (C, E) and three are not (A, B [7], D [31]). Colored circles and arcs in the figure represent corresponding graph nodes connected by temporal n-links. Ellipses represent super-pixel nodes[20]. Static n-links connect nodes with the same spatial location. Dynamic n-links connect nodes as a function of the image contents. In order to increase the robustness of approaches which are based on temporal link estimation, we ran the local graphcut algorithm in a forward-backward manner as follows: We initialized the forward pass by manually segmenting the first frame in the sequence and then ran local graphcuts in the forward direction. We then manually segmented the last frame in the sequence and ran the local graphcut approach in the backward direction. The final pixel-level segmentation output was the AND of the masks from the forward and backward passes.

The summary of quantitative results is shown in Table 2. From the table, we can see that using explicit motion estimation to instantiate the temporal links in the volume MRF is beneficial in comparison to a fixed topology. In subsequent sections, we refer to method (E), which had the best overall performance, as the baseline volumetric graph cut method.

We note that condition (D) resulted in the worst performance in our experiments. We hypothesize that this is due to the fact that when a super-pixel is assigned the wrong label, the error applies to all pixels and is therefore amplified by the area of the super-pixel. In the absence of human input to interactively modify the segmentation, this approach does not appear to lend itself to achieving high per-pixel segmentation accuracy.

We also point out that for the *monkey* sequence, all of the more complicated methods failed to perform as well as the simplest fixed topology temporal graph. We hypothesize that this was because of the reflection in the water and other challenging properties of this video, which make tracking and other more complex analysis more difficult.

5.2 Quantitative Comparison

We used the *GT-SegTrack database* to perform quantitative performance comparisons to three alternative tracking methods: the state-of-the-art level set-

sequence	A	B	C	D	E
cheetah	999	1116	891	1459	**874**
girl	3539	2919	2793	3364	**2217**
soldier	3021	1576	1394	4841	**1375**
monkey	5580	**2435**	3384	8726	3459

Table 2 Results of different temporal links: This table reports average number of error pixels per frame in four video sequences under five experimental treatments. Each treatment corresponds to the temporal n-link structure depicted in Figure 4. Treatment A corresponds to a graph without temporal n-links. The minimum error is highlighted in bold. Note that condition E results in minimum error for all sequences but *monkey*.

based tracker described in [15], the Rotobrush tool within Adobe AfterEffects CS5 (which seems to be based in part on [46]), and the baseline graphcut approach using KLT-based temporal links, which was described in section 3.5. In order to evaluate the level-set method [15], we carefully tuned the system to obtain the best results we could achieve, using source code which was generously provided by the authors. In each sequence, the target contour was initialized manually in the first frame. In our evaluation of Rotobrush, we used the interface in Adobe AfterEffects CS5 to manually segment the first frame by hand. We then ran the brush tool through the entire video sequence without any additional human input. Likewise for the baseline method.[4]

A quantitative comparison of tracking performance is provided in Table 3. Our per-pixel segmentation accuracy is better than that of all competing methods across most sequences. The only exception is the *parachute* sequence, where the baseline graphcut method performs slightly better. The large improvements in error which were observed for the *penguin* sequence are the result of tracker failure occurring in competing methods, with the result that the target contour vanished completely. In the case of the Rotobrush tool, it would be straight-forward to obtain more accurate segmentations through additional human intervention, by means of the excellent user-interface. While this experiment does not employ the Rotobrush tool in the manner in which it was designed to be used, it does provide a useful assessment of the ability of a state-of-the-art commercial product to perform automatic segmentation.

The main difference between our approach and the baseline methods described in section 5.1 is that our approach performs a joint optimization of segmentation and motion, while the other methods use explicit motion estimation to provide inputs to the segmentation process. The drawback of doing motion estimation and segmentation in separate stages is that in some challenging sequences the motion analysis might perform poorly, resulting in inaccurate inputs to the segmentation process. By incorporating the motion estimation step into the label space, our method has the opportunity to jointly-optimize both quantities.

[4] Note that for the experiments in this section, all algorithms were run in the usual "forward" manner, by segmenting the first frame and processing the subsequent frames in order.

sequence	Our score	[15] score	Rotobrush	Volume Graphcut
parachute	235	502	2441	**213**
girl	**1304**	1755	5450	1566
monkeydog	**563**	683	1178	726
penguin	**1705**	6627	3916	7041
birdfall	**252**	454	444	304
cheetah	**1142**	1217	1581	2183

Table 3 Scores on GT-SegTrack database: Scores correspond to average number of error pixels per frame. Select frames from *parachute*, *girl*, *monkeydog* and *birdfall* are illustrated in Figure 6, while frames from *penguin* are displayed in Figure 5. The minimum error is highlighted in bold.

sequence	Max	Min	Average
parachute	1.1515	1.1060	1.1341
girl	1.2134	1.0445	1.1338
monkeydog	1.3026	1.0829	1.1873
penguin	1.1574	1.0721	1.1241
birdfall	1.2388	1.0356	1.1452
cheetah	1.1635	1.1269	1.1486

Table 4 Suboptimality bound on GT-SegTrack database: We ran Fast-PD on each sliding window for every sequence. We report the range and average of the suboptimality bounds for each sequence.

5.3 Qualitative Comparison

Figure 5 provides a qualitative comparison between our method and those of [2], [15], and the Adobe AfterEffects Rotobrush tool, for select frames in five different test sequences. In order to compare our output with that of [2], since we lacked access to the authors' source code and segmentation masks, we identified the source video clips used in their experiments and applied our tracker to the same footage. Representative frames were extracted from their output videos and compared to identical frames from our output. In general, our method was able to provide more accurate segmentations, at the cost of additional computational resources in an off-line setting.

In Figure 6, we present additional segmentation results from our method on a variety of additional video sequences. The upper four sequences are a selection of frames from our GT-SegTrack database, while the last two clips are longer sequences from BBC's Planet Earth video series. Full length outputs are provided on our project website.

Optimality: In order to test the effectiveness of Fast-PD in this specific application, we report the suboptimality bounds for the algorithm on the GT-SegTrack database in Table 4. Compared with the bounds obtained for single image applications [29], the gaps are slightly larger. This might be due to the joint label space formulation and complexity of the video object segmentation. However, the average bounds are still close to 1 in all sequences, showing good performance.

Time and space requirements: Our algorithm was implemented in C++ and executed on a 2.66 Ghz Intel Core i5 processor with 24GB of memory. We measured its execution speed and memory requirements for a video resolution of 400 by 300 pixels, with the parameter settings provided in the paper. The computational cost of solving for a single sliding window position can be broken down into the cost of constructing the MRF model and the cost of optimizing it. Model construction was parallelized with OpenMP. It took 15 seconds when utilizing four cores. The optimization step ran in a single thread and took approximately 25 seconds per sliding window. The algorithm consumed approximately 20GB of memory at its peak.

6 Conclusion

We have described an off-line method for target tracking through the sequential segmentation of the video volume. Our formulation uses multi-label MRF optimization with an energy function that enforces spatio-temporal coherence. We present a ground-truth dataset for target tracking, called GT-SegTrack, which is based on a systematic assessment of the sources of difficulty in accurate segmentation. We compare our method to two recent on-line trackers, and demonstrate improved performance. Our results suggest that it is possible to obtain more accurate segmentations using an off-line approach, at the cost of increased computation. Our dataset is available from our project website.[5]

Acknowledgements Portions of this research were supported in part by NSF Grants 0916687 and 0960618, ONR HUNT MURI, and by Google Research. We would like to thank the authors of [15] for sharing their code with us. We would also like to thank Shaunak Vaidya and Vinit Patankar for their help with the GT-SegTrack database.

References

1. Balch, T., Dellaert, F., Feldman, A., Guillory, A., Isbell Jr., C.L., Khan, Z., Pratt, S.C., Stein, A.N., Wilde, H.: How multirobot systems research will accelerate our understanding of social animal behavior. Proceedings of the IEEE **94**(7), 1445–1463 (2006). Invited paper
2. Bibby, C., Reid, I.: Robust real-time visual tracking using pixel-wise posteriors. In: ECCV (2008)
3. Bluff, L., Rutz, C.: A quick guide to video-tracking birds. Biology Letters **4**, 319–322 (2008)
4. Bouguet, J.Y.: Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. Tech. rep., Microprocessor Research Labs, Intel Corp. (2002)
5. Boykov, Y., Funka-Lea., G.: Graph cuts and efficient n-d image segmentation. International Journal of Computer Vision (IJCV) **70**(2), 109–131 (2006)
6. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In: ICCV (2001)
7. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. on Pattern Analysis and Machine Intelligence **23**(11), 1222–1239 (2001)

[5] http://cpl.cc.gatech.edu/projects/SegTrack

8. Branson, K., Robie, A., Bender, J., Perona, P., Dickinson, M.: High-throughput ethomics in large groups of drosophilia. Nature Methods (2009)
9. Brostow, G., Essa, I., Steedly, D., Kwatra, V.: Novel skeletal representation for articulated creatures. ECCV (2004)
10. C. Rother V. Kolmogorv, A.B.: Grabcut: Interactive foreground extraction using iterated graph cuts. SIGGRAPH '04: ACM SIGGRAPH 2004 Papers **23**(3), 309–314 (2004)
11. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. IJCV **22**(1), 61–79 (1997)
12. Cham, T.J., Rehg, J.M.: A multiple hypothesis approach to figure tracking. CVPR (1999)
13. Chang, M.M., Tekalp, A.M., Sezan, M.I.: Simultaneous motion estimation and segmentation. IEEE Transactions on Image Processing **6**(9), 1326–1333 (1997)
14. Chellappa, R., Ferryman, J., Tan, T. (eds.): 2nd Joint IEEE Intl. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS 05). Beijing, China (2005). Held in conjunction with ICCV 2005
15. Chockalingam, P., Pradeep, N., Birchfield, S.: Adaptive fragments-based tracking of non-rigid objects using level sets. In: Intl. Conf. on Computer Vision (ICCV) (2009)
16. Dankert, H., Wang, L., Hoopfer, E.D., Anderson, D.J., Perona, P.: Automated monitoring and analysis of social behavior in drosophila. Nature Methods (2009)
17. Delcourt, J., Becco, C., Vandewalle, N., Poncin, P.: A video multitracking system for quantification of individual behavior in a large fish shoal: Advantages and limits. Behavior Research Methods **41**(1), 228–235 (2009). URL http://hdl.handle.net/2268/6100
18. Donoser, M., Bischof, H.: Fast non-rigid object boundary tracking. In: Proc. British Machine Vision Conf. (BMVC), pp. 1–10 (2008)
19. Felzenschwalb, P.: Representation and detection of deformable shapes. IEEE Trans. on Pattern Analysis and Machine Intelligence **27**(2), 208–220 (2005)
20. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV **59**(2) (2004)
21. Glocker, B., Paragios, N., Komodakis, N., Tziritas, G., Navab, N.: Inter and intra-modal deformable registration: continuous deformations meet efficient optimal linear programming. In: IPMI (2007)
22. Glocker, B., Paragios, N., Komodakis, N., Tziritas, G., Navab, N.: Optical flow estimation with uncertainties through dynamic MRFs. In: CVPR (2008)
23. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. CVPR (2010)
24. Kao, E.K., Daggett, M.P., Hurley, M.B.: An information theoretic approach for tracker performance evaluation. In: ICCV (2009)
25. Khan, Z., Balch, T., Dellaert, F.: Mcmc-based particle filtering for tracking a variable number of interacting targets. PAMI (2005)
26. Kohli, P., Torr, P.: Effciently solving dynamic markov random fields using graph cuts. In: ICCV, pp. 922–929 (2005)
27. Komodakis, N., Paragios, N., Tziritas, G.: MRF optimization via dual decomposition: Message-passing revisited. In: Intl. Conf. on Computer Vision (ICCV) (2007)
28. Komodakis, N., Tziritas, G.: A new framework for approximate labeling via graph cuts. In: ICCV (2005)
29. Komodakis, N., Tziritas, G.: Approximate labeling via graph-cuts based on linear programming. PAMI (2007)
30. Lempitsky, V., Boykov, Y.: Global optimization for shape fitting. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1–8 (2007)
31. Li, Y., Sun, J., Shum, H.Y.: Video object cut and paste. ACM Trans. Graph. **24**(3), 595–600 (2005)
32. Martin, J.: A portrait of locomotor behaviour in *Drosophila* determined by a video-tracking paradigm. Behavioural Processes **67**, 207–219 (2004)
33. Price, B.L., Morse, B.S., Cohen, S.: Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In: ICCV (2009)
34. Ramanan, D., Forsyth, D.: Using temporal coherence to build models of animals. Intl. Conf. on Computer Vision (ICCV) (2003)

35. Ren, X., Malik, J.: Tracking as repeated figure/ground segmentation. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2007)
36. Rodriguez, M.D., Ahmed, J., Shah, M.: Action mach: A spatio-temporal maximum average correlation height filter for action recognition. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2008)
37. Schoenemann, T., Cremers, D.: A combinatorial solution for model-based image segmentation and real-time tracking. IEEE Trans. on Pattern Analysis and Machine Intelligence **32**(7), 1153–1164 (2010)
38. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 593–600 (1994)
39. Sigal, L., Balan, A., Black, M.J.: Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. Intl. Journal of Computer Vision (2009)
40. Sminchisescu, C., Triggs, B.: Kinematic jump processes for monocular 3d human tracking. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 69–76 (2003)
41. Tsai, D., Flagg, M., Rehg, J.M.: Motion coherent tracking with multi-label mrf optimization. British Machine Vision Conference (BMVC) (2010). Recipient of the Best Student Paper Prize
42. Tsibidis, G., Tavernarakis, N.: Nemo: A computational tool for analyzing nematode locomotion. BMC Neuroscience **8**(1), 86 (2007). DOI 10.1186/1471-2202-8-86. URL http://www.biomedcentral.com/1471-2202/8/86
43. Vaswani, N., Tannenbaum, A., Yezzi, A.: Tracking deforming objects using particle filtering for geometric active contours. IEEE Trans. PAMI **29**(8), 1470–1475 (2007)
44. Wang, J., Bhat, P., Colburn, R.A., Agrawala, M., Cohen, M.F.: Interactive video cutout. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pp. 585–594. ACM, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1186822.1073233
45. Wang, P., Rehg, J.M.: A modular approach to the analysis and evaluation of particle filters for figure tracking. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 790–797. New York, NY (2006)
46. X.Bai, J.Wang, D.Simons, G.Sapiro: Video snapcut: Robust video object cutout using localized classifiers. In: SIGGRAPH (2009)
47. Xiao, J., Shah, M.: Motion layer extraction in the presence of occlusion using graph cuts. IEEE Trans. Pattern Analysis and Machine Intelligence **27**(10), 1644–1659 (2005)
48. Zhaozheng, Y., Collins, R.: Shape constrained figure-ground segmentation and tracking. In: CVPR (2009)
49. Zitnick, C.L., Jojic, N., Kang, S.B.: Consistent segmentation for optical flow estimation. In: ICCV (2005)

Fig. 5 Comparative results: Tracking results are illustrated for selected frames from five sequences, comparing our method to that of [2], [15] and Rotobrush.

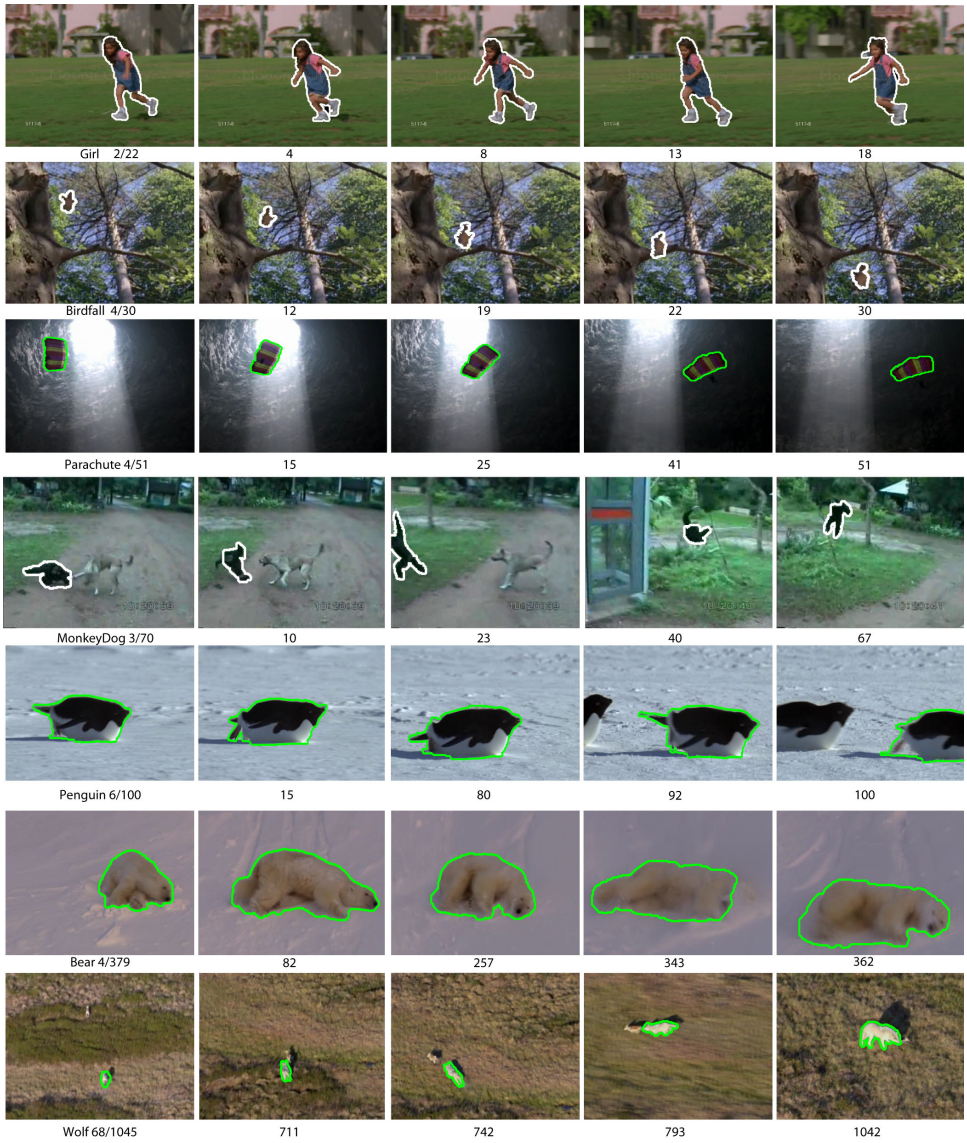

Fig. 6 Qualitative tracking results: Top: *Girl* sequence[36] from the UCF action database, illustrating shape changes. Row 2: Birdfall sequence from GT-SegTrack, exhibiting color overlap, large motion and small shape change, followed by Parachute, the easiest sequence in GT-SegTrack. Row4: Monkeydog sequence from SegTrack, showing large motion and significant shape change. Row5: One more penguin example. Rows 6 and 7: Two successfully tracked long sequences.