Interactive Tracking of 2D Generic Objects with Spacetime Optimization

Xiaolin K. Wei and Jinxiang Chai

Texas A&M University
xwei@cs.tamu.edu, jchai@cs.tamu.edu

Abstract. We present a continuous optimization framework for interactive tracking of 2D generic objects in a single video stream. The user begins with specifying the locations of a target object in a small set of keyframes; the system then automatically tracks locations of the objects by combining user constraints with visual measurements across the entire sequence. We formulate the problem in a spacetime optimization framework that optimizes over the whole sequence simultaneously. The resulting solution is consistent with visual measurements across the entire sequence while satisfying user constraints. We also introduce prior terms to reduce tracking ambiguity. We demonstrate the power of our algorithm on tracking objects with significant occlusions, scale and orientation changes, illumination changes, sudden movement of objects, and also simultaneous tracking of multiple objects. We compare the performance of our algorithm with alternative methods.

1 Introduction

Our objective in this paper is to track generic moving objects, including 2D locations, 2D scales and orientation, from a monocular video sequence. Building an interactive, accurate object tracking system is challenging because appearances of objects might change overtime due to occlusions, object deformations and illumination changes. Background clutter and sudden movements of the camera or object might further deteriorate the performance of a tracking system.

One solution to this problem is sequential object tracking, which initializes a tracker in one frame and then performs tracking forward recursively in time [1,2]. The approach is computationally efficient for online applications but might not be appropriate for many offline applications such as video-based motion capture, object-based video annotation, compression, and editing, where tracking accuracy is more preferred than real time performance. We believe an ideal system for offline object tracking should take full use of image measurements. When tracking goes wrong, user interaction is also needed to correct errors. Therefore, another desirable feature for offline tracking is to allow the user to specify constraints throughout the video to remove tracking ambiguities. The "feed forward" approach [1,2], however, does not consider frames and user constraints in the future.

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part I, LNCS 5302, pp. 657–670, 2008.

Another solution is to formulate offline tracking in a spacetime optimization framework [3] and consider the entire motion simultaneously [4,5,6]. The approach is appealing because the system can use image measurements and user constraints in both past and future to estimate the current state. Previous work in this direction often use discrete optimization to compute the optimal solution. However, as the number of frames or dimensions of the object state increase, discrete optimization might be too expensive to be conducted. Therefore previous approaches either use expensive preprocessing [4] or an object detector trained offline [5,6] to reduce the search space.

In this paper, we present a continuous trajectory optimization framework for offline tracking of 2D generic objects. Our optimization is very efficient; the speed of the spacetime tracker is comparable to that of sequential object tracking such as meanshift [1]. The system also does not require any preprocessing and offline learning steps. Other benefits of the framework are automatic occlusion handling with robust statistics, explicit modeling of illumination changes with weighted template models, ambiguity reduction with motion priors, and simultaneous tracking of multiple object with group priors.

We demonstrate the power and flexibility of this approach by interactive tracking of cars or sports players in various difficult video sequences. We show the system can accurately track a 2D generic object or or a group of objects with a minimal amount of user interaction. We also show its robustness to occlusions, background clutter, lighting and scale changes. Finally, we compare alternative techniques for 2D object tracking, including mean shift [2] and particle filter [1]. The experiment shows that the spacetime tracker can produce more accurate results with comparable processing time as sequential trackers.

2 Background

In this section, we discuss related work in tracking 2D generic objects from video. Previous work in 2D object tracking can be classified into three main categories: recursive tracking [2,7,8,9,1,10,11,12,13,14], batch-based optimizations [15,4], and tracking-by-detection [16,5,6].

Recursive object tracking methods initialize one frame and then recursively track the evolution of the state forward in time. For example, kernel-based methods sequentially track the state of nonrigid objects by minimizing viewpoint-insensitive histograms between consecutive frames [2,7,13,14]. Kalman filter extensions achieve more robust tracking of maneuvering objects by introducing statistical models of object or camera motion [8,9]. Tracking through occlusion and clutter is achieved by reasoning over a state-space of multiple hypotheses via sequential monte carlo sampling methods [1,10,12].

In contrast, batch-based optimization approach formulates the tracking problem as a trajectory optimization and computes the entire motion sequence simultaneously. In particular, Sun and his colleagues explored discrete optimization to track the object state (2D positions and 1D uniform scale) throughout the video [4]. To reduce the search space for discrete optimization, they first run

the mean shift algorithm to identify 2D candidate regions of the object in each frame, and then applied spectral clustering to extract a number of 3D trajectory segments of the object. Our approach also uses batch-based optimization for 2D object tracking. Unlike previous approaches, we formulate the problem in a continuous optimization framework. The continuous optimization allows us to simultaneously track a group of objects, a capacity that has not been demonstrated in discrete optimization approaches. More importantly, our continuous optimization does not require any preprocessing steps.

Another approach for 2D object tracking is to apply object detection algorithms for tracking. For example, researchers have adapted SVM recognition algorithms for efficient visual tracking [16]. Recently, detection and optimization have also been combined to obtain some of the advantages of each approach [5,6]. However, detection approaches require an offline training step for particular objects, and might not be appropriate for tracking an arbitrary 2D objects.

Our approach is motivated by keyframe guided rotoscoping [15] which uses continuous optimization to track contours of an object with user-specified contours in two or more keyframes. Rotoscoping makes full use of the information in the keyframes to improve the performance of contour tracking. Our work extends this approach significantly by applying it to 2D generic objects.

3 Overview

We formulate object tracking as a spacetime optimization problem, which optimizes over entire sequence simultaneously under user-specified constraints. The system contains three major components:

User interactions. The user begins by clicking the location of target objects in the first and the last frame. The keyframes are then used to interpolate inbetween motions based on image measurements of the entire sequence and predefined priors.

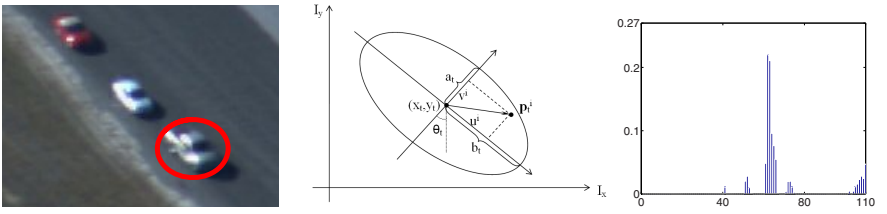
Spacetime optimization. The system computes the "best" motion by optimizing over the whole image sequence at once. The objective function includes keyframe constraints, a weighted template model, an image measurement term and motion prior terms.

Refinement. Because of the difficulty of the tracking problem, there will often be unsatisfactory aspects of the tracking results. Once the optimization has completed, the user may refine tracking results in any frame and then rerun the optimization with these new constraints.

We define the feature model of target objects in the next section and then discuss the spacetime tracking in detail in section 5.

4 Target Representation

In this section, we first discuss how to define a feature space to characterize target objects (section 4.1). We then describe how to represent the template of



(a) (b) (c)

Fig. 1. Target representations: (a) A target object is approximated as an elliptic region; (b) The elliptic region is parameterized by 5-dimensional vector $\mathbf{z}_t = (x_t, y_t, a_t, b_t, \theta_t)^T$; (c) The appearance of the object can thus be represented using histogram distributions of color information in the HSV color space

target objects with weighted template models (section 4.2). We also show how to evaluate the distance between the target template and target candidate (section 4.3).

4.1 Feature Space

In our experiments, a target object is approximated by an elliptic region (see Figure 1(a)). The state of the target object at frame t can thus be described by parameters of the ellipse (see Figure 1(b)):

$$\mathbf{z}_t = (x_t, y_t, a_t, b_t, \theta_t)^T \tag{1}$$

where the parameters x_t and y_t represent the coordinates of the center of the ellipse and the parameters a_t and b_t specify the lengths of the long and short axes respectively. The parameter θ_t is the orientation of the ellipse.

We choose the Hue-Saturation-Value (HSV) color space to model the appearance of target objects. We further define the feature model of a target object as a histogram distribution of all pixels within the elliptic region (see Figure 1(c)). The feature model of a target object at frame t is represented by

$$\mathbf{h}(\mathbf{z}_t) = (h_1(\mathbf{z}_t), \ldots, h_M(\mathbf{z}_t))^T, \quad \sum_{m=1}^{M} h_m(\mathbf{z}_t) = 1 \tag{2}$$

where the parameter M is the total number of bins used in the HSV color space and the function $h_m(\mathbf{z}_t)$ is the density of the m-th bin. Let n_t be the number of pixels located inside the target region at frame t and $\mathbf{p}_t^i, i = 1, ..., n_t$ be the image coordinates of the i-th pixel. Mathematically, we can define the function $h_m(\mathbf{z}_t)$ as follows:

$$h_m(\mathbf{z}_t) = \sum_{i=1}^{n_t} \delta(f(I(\mathbf{p}_t^i)) - m) \tag{3}$$

where the function $\delta(\cdot)$ represents the Kronecker delta function. The function $I(\mathbf{p}_t^i)$ represents the color of the i-th pixel at the location $\mathbf{p}_t^i$. The function f maps $I(\mathbf{p}_t^i)$ to the index of its bin in the quantized feature space.

Because color information is only reliable when both the saturation and the value are not too small, we populate an HS histogram with $N_h N_s$ bins using

only the pixels with saturation and value larger than 0.1 and 0.2 respectively. The remaining "color-free" pixels can however retain a crucial information when tracked regions are mainly black and white. We thus use N_v bins to quantize the V space separately. The resulting complete histogram is composed of $M = N_h N_s + N_v$ bins. In our experiments, N_h, N_s and N_v are set to 10 experimentally. Figure 1(c) shows a histogram distribution computed from color information of a target object shown in Figures 1(a).

We regularize the histogram distribution $h_m(\mathbf{z}_t)$ by masking the objects with an isotropic kernel in the spatial domain. When the kernel weights, carrying continuous information, are used in defining the feature space representation, the regularized histogram distribution of target regions becomes a smooth and continuous function of target states, $\mathbf{z}_t$.

An isotropic kernel, with a convex and monotonic decreasing kernel function $k(r)$, is used to assign smaller weights to pixels farther from the center. In our experiments, we choose the Epanechnikov profile as our kernel function [17]:

$$k(r) = \begin{cases} 1 - r & 0 \le r \le 1 \\ 0 & r > 1 \end{cases} \tag{4}$$

where $r \ge 0$. This kernel function makes the regularized histogram distribution differentiable everywhere inside the elliptical region. Its gradients can, therefore, be evaluated analytically.

The regularized histogram distribution of the feature in the target region at frame t is computed as:

$$h_m(\mathbf{z}_t) = \frac{\sum_{i=1}^{n_t} k\left((\frac{u^i}{a})^2 + (\frac{v^i}{b})^2\right) \delta\big(f(I(\mathbf{p}_t^i)) - m\big)}{\sum_{i=1}^{n_t} k\left((\frac{u^i}{a})^2 + (\frac{v^i}{b})^2\right)} \tag{5}$$

The parameters u^i and v^i are the local coordinates of the i-th pixel (see Figure1(b)), which can be computed as follows:

$$\begin{pmatrix} u^i \\ v^i \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} (\mathbf{p}_t^i - \mathbf{c}_t) \tag{6}$$

where the vector $\mathbf{c}_t$ is a 2D vector (x_t, y_t). Therefore, the feature model of a target region $\mathbf{h}(\mathbf{z}_t)$ is a vector-valued function of the object state $\mathbf{z}_t = (x_t, y_t, a_t, b_t, \theta_t)^T$.

4.2 Weighted Template Model

Our interactive tracking system starts with keyframe constraints defined at the first and last frame. Let $\mathbf{z}_1$ and $\mathbf{z}_T$ represent the state of the first and last frame respectively. We start by computing the feature models of the first and last frame $\mathbf{h}(\mathbf{z}_1)$ and $\mathbf{h}(\mathbf{z}_T)$.

We assume that the template model for any in-between frames can be modeled as a linear interpolation of the feature models at keyframes:

$$H_m(\beta_t) = \beta_t h_m(\mathbf{z}_1) + (1 - \beta_t) h_m(\mathbf{z}_T) \qquad 0 \le \beta_t \le 1,\ 1 < t < T \tag{7}$$

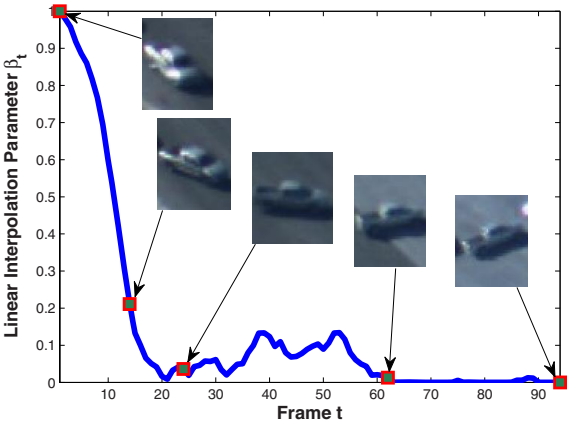


Fig. 2. The optimized β_t over time for the "Car Team" sequence: The red points are the optimized β values for five frames shown on the top respectively

where the interpolation weight β_t is a scalar value between 0 and 1. The interpolated feature model at frame t, $\mathbf{H}(\beta_t) = (H_1(\beta_t), \ldots, H_M(\beta_t))^T$, is therefore a function of β_t. One attractive feature of the weighted template models is to model possible appearance changes between two keyframes with the parameter β_t. For example, when the weight β_t is close to one, the appearance of the template object becomes similar to that of the first keyframe. When the weight β_t approaches to zero, the template object looks more like the target object in the last keyframe.

Figure 2 shows the evolution of the weight β_t optimized from one testing video sequence. The images on the top show five sample frames of the target vehicle. The appearances of the target object is consistent with the reconstructed β_t values (red points).

4.3 Similarity Function

The similarity function is used to evaluate the distance between a template object and candidate objects in the feature space.

We define their distance, d, as

$$d(\mathbf{z}_t, \beta_t) = \sqrt{1 - \sum_{m=1}^{M} \sqrt{h_m(\mathbf{z}_t)\, H_m(\beta_t)}} \tag{8}$$

where the function $h_m(\mathbf{z}_t)$ is the feature model of an candidate object and the function $H_m(\beta_t)$ is the weighted template model at frame t. Both functions are regularized histogram distributions. A differential kernel function yields a differentiable similarity function. We, therefore, can evaluate the gradient of the similarity function in terms of β_t and $\mathbf{z}_t$ analytically.

5 Spacetime Object Tracking

We formulate object tracking as a continuous trajectory optimization problem and estimate the states of objects across the entire sequence simultaneously. This section discusses how to derive the objective function and how to optimize it efficiently.

5.1 Objective Function

The objective function consists of two types of energy terms: *image* term and *prior* term. The image term prefers a solution which minimizes appearance difference between the template object and candidate object. The prior term penalizes quickly moving objects and sudden changes of scales and appearances. The prior term is pivotal for removing the tracking ambiguity due to occlusions or clutter.

Image Term. The data term, E_I, measures similarity between the template model $\mathbf{H}(\beta_t)$ and candidate object $\mathbf{h}(\mathbf{z}_t)$ ranging from frame 2 to frame $T-1$.

To deal with occlusions, we apply robust statistics to measure the data term. Robust estimation addresses the problem for finding the values for the parameters from the measurement data with outliers, which correspond to heavily occluded objects in our experiments. We therefore can define the data term as follows:

$$E_I = \sum_{t=2}^{T-1} \rho(d(\mathbf{z}_t, \beta_t)) \tag{9}$$

To increase robustness we will consider estimators for which the influence of outliers tends to zero. We choose the Lorentzian estimator but the treatment here could equally be applied to a wide variety of the other estimator. A discussion of various estimators can be found in [18,19]. More specifically, the Lorentzian function is defined as follows:

$$\rho(r) = \log(1 + 1/2(r/\sigma)^2) \tag{10}$$

where the scalar σ is a parameter for the robust estimator. In our experiments, σ is set to 0.2475.

Prior Terms. Due to occlusions, background clutter, illumination or scale changes, image term might not be sufficient for tracking objects correctly. Prior term is very important for reducing the tracking ambiguity. There are two types of prior terms used in our paper:

- The state smoothness term, E_P, minimizes state changes over time:

$$E_P = \sum_{t=2}^{T} \|\mathbf{z}_t - \mathbf{z}_{t-1}\|^2 \tag{11}$$

  where $\mathbf{z}_t$ is the state of the target object at frame t, which includes 2D location, 2D scales, and orientation.

– The illumination smoothness term, E_L, measures the temporal smoothness of the interpolation weight, β_t, over time.

$$E_L = \sum_{t=2}^{T} (\beta_t - \beta_{t-1})^2 \tag{12}$$

Tracking objects only by prior terms is similar to performing smooth interpolation of keyframes.

5.2 Optimization Method

After combining the image term and prior terms, our interactive object tracking problem becomes the following constrained nonlinear optimization problem:

$$\begin{array}{c} arg\, min_{\mathbf{Z},\beta} \;\; E_I(\mathbf{Z},\beta) + \lambda_P E_P(\mathbf{Z}) + \lambda_L E_L(\beta) \\ s.t. \;\; 0 \le \beta_t \le 1 \;\; t = 1, ..., T \end{array} \tag{13}$$

where $\mathbf{Z}$ and β are the concatenation of the system states $\mathbf{z}_t$ and the concatenation of the interpolation weights β_t from frame 2 to frame $T-1$. The weights λ_P and λ_R are set to 0.0015 and 0.5 respectively.

The number of variables in our optimization can be quite large — about five times the number of total frames. The Jacobian matrix for our objective function, however, is very sparse. We therefore optimize the objective function with large-scale trust-region reflective Newton methods.

A linear interpolation of the user-specified keyframes is used for optimization initialization. We found that the optimization procedure runs very efficiently and always converges quickly (usually less than 20 iterations). In our experiments, we found that the spacetime tracker with a Matlab implementation can track objects interactively (from 12 fps to 25 fps).

The gradient of the energy function is analytically evaluated at each iteration (See Appendix). The two prior terms, Equation 11 and Equation 12, are quadratic functions, whose derivatives can be easily derived. The partial derivatives of the image term can also be evaluated analytically because the similarity function is analytically differentiable (See Appendix).

Due to the complexity of a real world, it is almost impossible to build a fully automatic system that can robustly track any interesting objects from video. When a tracker fails, user interaction must be used to correct tracking errors. Our system provides an efficient way to combine user interactions with an automatic vision process. The user can refine the tracking result at any frame and restart the optimization. When more than two keyframes are defined, the spacetime solver computes the solution for each subsequence separately.

Like any gradient-based optimization methods, the system might fall in local minima. Though, we rarely have this problem in our experiments. One strength of the system is to involve the user into the loop. If the optimization falls in a local minima, the user can briefly review the result and then specify more keyframes to rerun the optimization.

Table 1. The statistics for video sequences tested in our paper

Sequence	Length	Occlusions	Scale changes	Lighting changes	Running Time
Touchdown	100	large	large	none	11.9s
IR	240	partial	small	large	14.9s
Rushing	94	large	small	none	9.4s
Truck in Forest	130	large	medium	large	22.4s
Car Teams (one car)	94	partial	small	medium	6.3s

#1 #45 #75 #90 #100

Fig. 3. The "Touchdown" sequence shows our algorithm's robustness to scale changes

5.3 Multi-object Tracking

Our framework can also be used to track a group of objects simultaneously. When multiple objects move as a group such as a group of racing cars or football players, their movements are not independent. One advantage of simultaneous tracking of multiple objects is to utilize motion prior between multiple objects to reduce tracking uncertainty.

Similarly, we can formulate the multi-object tracking problem as the following optimization problem:

$$argmin_{\{\mathbf{Z}^n,\beta^n\}} \sum_n (E_I^n(\mathbf{Z}^n,\beta^n)+\lambda_P E_P(\mathbf{Z}^n) + \lambda_L E_L(\beta^n)) + \lambda_G E_G(\mathbf{Z}^1,...,\mathbf{Z}^N)$$
$$s.t. \quad 0 \leq \beta_t^n \leq 1 \quad t = 1,...,T \tag{14}$$

where N is the total number of objects to be tracked. The parameters $\mathbf{Z}^n$ and β^n are the concatenation of the system states $\mathbf{z}_t$ and the concatenation of the interpolation weights β_t over time for the n-th object. The optimization term E_G models group prior for multi-object movements. In our experiments, we choose to minimize the changes of the relative distances between multiple objects, though more sophisticated group priors might be used.

6 Experimental Results

In this section, we first test the performance of our algorithm on real video sequences. We then compare our algorithm with two alternative object tracking techniques–particle filter [1] and mean shift [2]. A short summary of testing video sequences and their computational time is reported in Table 1. All the computational time reported here is based on Matlab implementations. Our tracking results are best seen in the accompanying video.

6.1 Testing on Real Video

We tested the effectiveness of our system on different kinds of video sequences with occlusions, scale changes, and illumination changes (see Figures 3, 4, 5, and 6). We have also done experiments on simultaneous tracking of multiple objects (see Figure 7).

The "Touchdown" sequence contains a target object with significant scale changes (see Figure 3). Throughout the testing sequence, the scales of the player increase from $(a_1, b_1) = (10.1, 22.5)$ to $(a_{100}, b_{100}) = (45.1, 117.7)$ because of the zooming of a camera.

The "IR" sequence has significant illumination changes. The low-resolution and noisy video makes the tracking problem even more challenging. Figure 4 shows our system can track the video accurately.

The "Rushing" sequence demonstrates the performance of our algorithm on tracking target objects with significant occlusions (see Figure 5). The video has significant occlusions in two different places. From frame 15 to 27, the target player is occluded by an opponent player in a different uniform. The second occlusion lasts longer (from frame 59 to 81); the target player is completely occluded by his teammate with the same uniform during a certain period of time.

The "Truck in Forest" sequence demonstrates the performance of our algorithm on tracking target objects with significant occlusions, illumination changes (shadows) and orientation changes (see Figure 6). In addition, the robust estimator for the data term allows the system to automatically detect when the object is heavily occluded (see blue ellipses).

Fig. 4. The "IR" sequence includes significant illumination changes and is successfully tracked by our algorithm

Fig. 5. The "Rushing" sequence includes significant occlusions and is successfully tracked by our algorithm

Fig. 6. The "Truck in Forest" sequence shows the performance of our algorithm on tracking an object with significant occlusions. The system can also detect when the target is occluded (in blue ellipses).

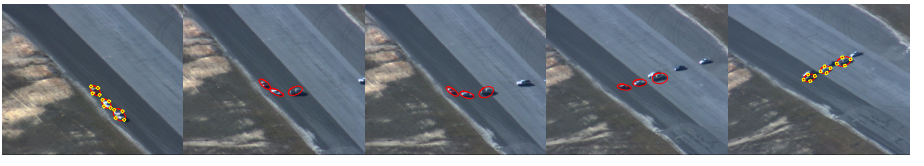


Fig. 7. Simultaneous tracking of three cars with group priors

The "Car Team" sequence demonstrates the power of our algorithm on tracking multiple objects simultaneously (see Figure 7). It is a difficult sequence to track. The cars have a sudden turn in the middle of the sequence and target objects also look very similar. With the group prior described in section 5.3, we successfully track the movements of three cars in the same team.

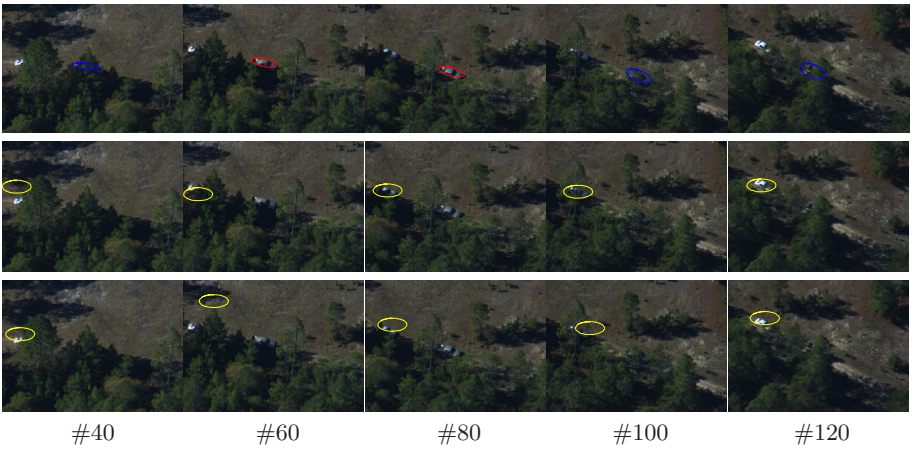


Fig. 8. Comparisons on the "Truck in Forest" sequence. Our spacetime tracker (first row) successfully tracked the whole sequence. The particle filter (second row) and the mean shift method (third row) failed to track the object across the entire sequence.

Table 2. The number of keyframes needed and the running time (in brackets) for different methods. The running time is based on Matlab implementations.

Sequence	Mean shift	Particle filtering	Our method
Touchdown	4 (214s)	5 (74s)	2 (12s)
IR	6 (31s)	5 (19s)	2 (15s)
Rushing	6 (33s)	6 (19s)	2 (9s)
Truck in Forest	9 (180s)	8 (52s)	2 (22s)
Car Teams (one car)	5 (49s)	4 (27s)	2 (6s)

6.2 Comparisons with Other Methods

This section compares the performance of our spacetime tracking algorithm with particle filter [1] and mean shift [2]. Figure 8 shows comparison results for the "Truck in Forest" sequence. Table 2 reports, for different methods, the number of keyframes and the computational time needed in order to track the targets successfully. Our method provides better performance than the two alternative algorithms.

7 Discussion

We have presented a spacetime optimization approach for 2D generic object tracking. Unlike previous offline tracking algorithms [4,5,6], our system uses continuous optimization for 2D object tracking and does not require any preprocessing or off-line learning learning steps. The system can achieve high quality results with minimal user input. Our experiments demonstrate the power of our algorithm on tracking objects with occlusions, scale changes, illumination changes, and sudden movement of objects. We also show the performance algorithm on simultaneous tracking of multiple objects which might be too expensive for previous offline tracking algorithms.

As compared to recursive tracking methods, our method is more appropriate for interactive applications such as object-based video annotation, compression, editing, and video based motion capture. For those applications, high quality results are far more important than real-time performances. In the mean time, all video frames are available in advance. Combining user interaction with image measurements across the entire video sequence provides more accurate results than automatic tracking methods.

We believe the basic idea of our spacetime tracking could also be used for tracking other types of objects. One of immediate directions for future work is, therefore, to extend our spacetime optimization for interactive feature tracking and articulated object tracking.

References

1. Isard, M., Blake, A.: Condensation conditional density propagation for visual tracking. International Journal on Computer Vision 29(1), 5–28 (1998)
2. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Trans. Pattern Analysis and Machine Intelligence 25(3), 564–577 (2003)

3. Witkin, A., Kass, M.: Spacetime constraints. In: Proceedings of ACM SIGGRAPH 1998, pp. 159–168 (1988)
4. Sun, J., Zhang, W., Tang, X., Shum, H.Y.: Bi-directional tracking using trajectory segment analysis. In: Proceedings of ICCV, vol. 1, pp. 717–724 (2005)
5. Buchanan, A., Fitzgibbon, A.: Interactive feature tracking using k-d trees and dynamic programming. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 626–633 (2006)
6. Wei, Y., Sun, J., Tang, X., Shum, H.Y.: Interactive offline tracking for color object. In: Proceedings of ICCV (2007)
7. Elgammal, A., Duraiswami, R., Davis, L.: Probabilistic tracking in joint feature-spatial spaces. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 781–788 (2003)
8. Blackman, S., Popoli, R.: Design and analysis of modern tracking systems. Artech House Publishers (1999)
9. Koller, D., Daniilidis, K., Nage, H.: Model-based object tracking in monocular image sequences of road traffic scenes. International Journal on Computer Vision 10(3), 257–281 (1993)
10. Rasmussen, C., Hager, G.: Joint probabilistic techniques for tracking multi-part objects. IEEE Trans. Pattern Analysis and Machine Intelligence 23, 560(n)-576 (1993)
11. Jepson, A., Fleet, D., El-Maraghi, T.: Robust online appearance models for visual tracking. IEEE Trans. Pattern Analysis and Machine Intelligence 25(10), 1296–1311 (2003)
12. Wu, Y., Huang, T.S.: Robust visual tracking by integrating multiple cues based on co-inference learning. International Journal on Computer Vision 58(1), 55–71 (2004)
13. Guskov, I.: Kernel-based template alignment. In: Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 610–617 (2006)
14. Megret, R., Mikram, M., Berthoumieu, Y.: Inverse composition for multi-kernel tracking. In: Computer Vision, Graphics and Image Processing. LNCS, pp. 480–491 (2007)
15. Agarwala, A., Hertzmann, A., Salesin, D.H., Seitz, S.M.: Keyframe-based tracking for rotoscoping and animation. ACM Transactions on Graphics 24(3), 584–591 (2005)
16. Avidan, S.: Support vector tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(8), 1064–1072 (2004)
17. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Analysis and Machine Intelligence 24(5), 603–619 (2002)
18. Huber, P.: Robust statistics. Wiley, Chichester (1981)
19. Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A.: Robust statistics: The approach based on influence functions. Wiley, Chichester (1986)

Appendix

This appendix shows how to derive the Jacobian matrix of Equation 8. The Jacobian matrix can be analytically evaluated based on the following partial derivatives.

$$\frac{\partial d(\mathbf{z}_t,\beta_t)}{\partial x_t} = -\frac{1}{a_t^2}F(\mathbf{z}_t,\beta_t)\sum_{m=1..M}^{h_m(\mathbf{z}_t)\neq 0}\sqrt{\frac{H_m(\beta_t)}{h_m(\mathbf{z}_t)}}\sum_{i=1}^{n_t}u_t^i\left(\delta_m(\mathbf{p}_t^i)-h_m(\mathbf{z}_t)\right) \tag{15}$$

$$\frac{\partial d(\mathbf{z}_t,\beta_t)}{\partial y_t} = -\frac{1}{b_t^2}F(\mathbf{z}_t,\beta_t)\sum_{m=1..M}^{h_m(\mathbf{z}_t)\neq 0}\sqrt{\frac{H_m(\beta_t)}{h_m(\mathbf{z}_t)}}\sum_{i=1}^{n_t}v_t^i\left(\delta_m(\mathbf{p}_t^i)-h_m(\mathbf{z}_t)\right) \tag{16}$$

$$\frac{\partial d(\mathbf{z}_t,\beta_t)}{\partial a_t} = -\frac{1}{a_t^3}F(\mathbf{z}_t,\beta_t)\sum_{m=1..M}^{h_m(\mathbf{z}_t)\neq 0}\sqrt{\frac{H_m(\beta_t)}{h_m(\mathbf{z}_t)}}\sum_{i=1}^{n_t}(u_t^i)^2\left(\delta_m(\mathbf{p}_t^i)-h_m(\mathbf{z}_t)\right) \tag{17}$$

$$\frac{\partial d(\mathbf{z}_t,\beta_t)}{\partial b_t} = -\frac{1}{b_t^3}F(\mathbf{z}_t,\beta_t)\sum_{m=1..M}^{h_m(\mathbf{z}_t)\neq 0}\sqrt{\frac{H_m(\beta_t)}{h_m(\mathbf{z}_t)}}\sum_{i=1}^{n_t}(v_t^i)^2\left(\delta_m(\mathbf{p}_t^i)-h_m(\mathbf{z}_t)\right) \tag{18}$$

$$\frac{\partial d(\mathbf{z}_t,\beta_t)}{\partial \theta_t} = (\frac{1}{a_t^2}-\frac{1}{b_t^2})F(\mathbf{z}_t,\beta_t)\sum_{m=1..M}^{h_m(\mathbf{z}_t)\neq 0}\sqrt{\frac{H_m(\beta_t)}{h_m(\mathbf{z}_t)}}\sum_{i=1}^{n_t}u_t^iv_t^i\left(\delta_m(\mathbf{p}_t^i)-h_m(\mathbf{z}_t)\right) \tag{19}$$

$$\frac{\partial d(\mathbf{z}_t,\beta_t)}{\partial \beta_t} = \frac{1}{4d(\mathbf{z}_t,\beta_t)}\sum_{m=1..M}^{H_m(\beta_t)\neq 0}\sqrt{\frac{h_m(\mathbf{z}_t)}{H_m(\beta_t)}}\left(H_m(\beta_t)-H_m(\beta_1)\right) \tag{20}$$

where,

$$F(\mathbf{z}_t,\beta_t) = \frac{1}{2d(\mathbf{z}_t,\beta_t)\sum_{j=1}^{n_t}k\left((u_t^i/a_t)^2+(v_t^i/b_t)^2\right)} \quad \text{and} \quad \delta_m(\mathbf{p}_t^i) = \delta(f(I(\mathbf{p}_t^i))-m)$$

Note that, in Equation 15, 16, 17, 18 and 19, we should drop off the components when $h_m(\mathbf{z}_t)=0$ in the summations over m. Because when $h_m(\mathbf{z}_t)=0$, $h_m(\mathbf{z}_t)$ will not change with respect to $\mathbf{z}_t$ (see Equation 5). Thus partial derivatives of $\sqrt{h_m(\mathbf{z}_t)H_m(\beta_t)}$ (in Equation 8) with respect to $\mathbf{z}_t$ are zeros, so the m's with $h_m(\mathbf{z}_t)=0$ should be dropped off. Similarly, the components with $H_m(\beta_t)=0$ in Equation 20 also need to be dropped off.

The distance function (Equation 8) is differentiable everywhere inside the elliptical region. Therefore the whole objective function is also differentiable inside the elliptical region.