2017 14th Conference on Computer and Robot Vision

Fully Automatic, Real-Time Vehicle Tracking for Surveillance Video

Yanzi Jin, Jakob Eriksson
University of Illinois at Chicago, Computer Science Dept.
{yjin25, jakob}@uic.edu

***Abstract*—We present an object tracking framework which fuses multiple unstable video-based methods and supports automatic tracker initialization and termination.**

To evaluate our system, we collected a large dataset of hand-annotated 5-minute traffic surveillance videos, which we are releasing to the community. To the best of our knowledge, this is the first publicly available dataset of such long videos, providing a diverse range of real-world object variation, scale change, interaction, different resolutions and illumination conditions.

In our comprehensive evaluation using this dataset, we show that our automatic object tracking system often outperforms state-of-the-art trackers, even when these are provided with proper manual initialization. We also demonstrate tracking throughput improvements of 5× or more vs. the competition.

I. Introduction

Vehicle tracking has important applications in traffic engineering. Over time, a number of vehicle counting methods have been developed, including specialized hand-held counting boards with buttons to push, pressure tubes laid across the pavement, magnetic loops under the pavement and more. Overall, the most powerful techniques rely on manual input and tend to be extremely labor intensive, whereas the mostly automatic techniques lack in accuracy and descriptiveness. In principle, computer vision provides the most scalable and economical alternative. Ideally, a fully automatic computer vision-based tracker follows each vehicle as it enters, traverses and exits the scene. However, current tracking algorithms such as [1], [2], [3], [4] all require initialization as input, leading to semi-automatic tracking systems. To avoid manual input, these trackers rely on background subtraction and/or object detectors for initialization. Here, the primary challenge is robustness to variations in illumination condition, viewpoint, and video quality. Background subtraction model could fail with illumination change, while detectors are not appropriate for detecting a vehicle in the distance, or in a grainy low-resolution video, as our experiments demonstrate. Additionally, the low throughput of most trackers prevents widely deployed surveillance applications, as VOT 2016 challenge reports that none of the top-ranked trackers run in real-time for even a single tracked object.

We propose a fully automatic algorithm for vehicle tracking that runs faster than real-time. With a sensor fusion approach, we combine background segmentation, object detection, and optical flow into a single, robust vehicle tracking system via Kalman filtering. Initialization uses the same three sources, to automatically identify moving objects in the scene. Finally, when an object exits the scene, its movements are analyzed to filter out unlikely object trajectories. To evaluate our algorithm as well as prior work, we create a hand-annotated dataset, consisting of 11 diverse, 5-minute videos collected from existing traffic surveillance cameras. For each frame, the location and extent of each moving object is provided, which enables accurate, quantitative evaluation.

We compare the proposed algorithm against multiple state-of-the-art trackers, which rely on human input for initialization. On this dataset, we report considerably better performance than the state of the art with manual initialization, and substantial accuracy improvement when using our new automatic initialization method. Moreover, we demonstrate throughput 4× faster than real time and over 5× improvement compared to 5 out of 7 several baseline trackers, up to 47×.

II. Preliminary

The problem of vehicle tracking in existing traffic surveillance video presents some unique computer vision challenges, including scale changes, video quality (exposure control, automatic white balance and compression), weather conditions, illumination changes, variations in perspective, and occlusion. Current work in object detection [5], [6], tracking [1], [3], and background subtraction [7], [8] can deal with a subset of these conditions, but so far a generic system has been elusive. Below, we first introduce the underlying methods used in our system, then describe our vehicle tracking framework in detail.

A. Background subtraction

Background subtraction generates a binary foreground mask given a sequence of frames. Connected areas in the foreground mask can be treated as moving objects, although this technique can be error prone. We use ViBe [7], for its balance of speed, robustness and accuracy. However, other methods could be substituted with acceptable results in many cases.

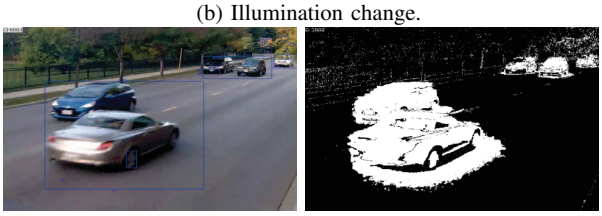
Fig. 1 summarizes four common failure cases of the background subtraction with foreground bounding boxes on the original frame on the left and the foreground on the right. Automatic exposure (Fig. 1(a)) is performed by the camera during recording, whereas illumination variation (Fig. 1(b)) is due to external light sources, such as the sun and vehicle headlights. Both automatic exposure and illumination variation cause rapid and widespread changes in pixel values, which most background subtraction methods struggle with. Occlusion (Fig. 1(c)) creates a single connected foreground area

978-1-5386-2818-8/17 $31.00 © 2017 IEEE
DOI 10.1109/CRV.2017.43

out of two or more moving objects, or sometimes multiple foreground areas for a single moving object, breaking any assumption of a one-to-one mapping between foreground areas and moving object. Ghosting (Fig. 1(d)) usually happens when a foreground object remains stationary for a long time, during which time it is gradually assimilated into the background model. When the object begins to move, what appears from behind the object is inaccurately marked as foreground, until the background model has had time to adjust.

(a) Camera auto-exposure.

(b) Illumination change.



(c) Occlusion. Note also the shadow, due to illumination change.

(d) Ghosting. Vehicles stopped at red light have become part of the background model.

Fig. 1: Common background subtraction failure cases. Pixel values change for many reasons other than motion.

In summary, background subtraction provides the ability to capture small movements without manual setup beforehand. However, it is error-prone and must be compensated by other methods to create a robust vehicle tracking system.

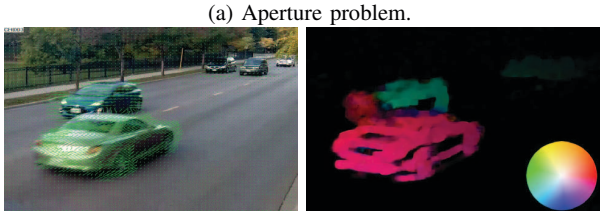
B. Object detection

Object detectors [5], [6] work on individual frames, scanning the image for areas that appear similar to offline training samples. Compared to background subtraction, object detection method tends to be more robust to illumination change and occlusion. However, the cost of object detection is remarkable, as it often involves an exhaustive search throughout the image, both in location and object size. Cascaded classifiers partially address this by discarding background regions [5]. More recently, deep neural networks [6], [9] have emerged as a promising approach to object detection. We use a state-of-the-art detector called faster-RCNN [9]. Running on a high-end graphics processing unit (GPU), the time required for detection on one image drops from 2 seconds [5] to 198 ms on the PASCAL 2007 dataset, making the real-time detection in video feasible. However, like other detectors, faster-RCNN still has missing and false detections. In our measurements, it has a missing rate in excess of 65% and 86% on high- and low-resolution videos, respectively. Thus, given the high miss rate, especially on poor quality images, object detection alone will not suffice for a robust vehicle tracking system.

C. Optical flow

Optical flow is an estimate of the movement of pixels between two images: in our case, two consecutive video frames. Optical flow provides a low-level description of motion in images and can offer useful evidence for tracking applications. Estimating optical flow is a research area in its own right, but we use the seminal Lucas-Kanade algorithm [10] in our system, as it runs fast on GPUs, and provides useful results while making minimal assumptions about the underlying scene and image. Fig. 2 illustrates two optical flow problems that may affect tracking accuracy. The left column shows the direction and magnitude of the optical flow vectors, while the right column is the color code visualization of the optical flow results, with the color wheel at bottom right corner of Fig. 2(b) indicating the corresponding direction. Fig. 2(a) illustrates the so-called aperture problem, where the center of the truck has no reported optical flow, due to its large and uniformly colored surface. Fig. 2(b) illustrates the "turbulent", error-prone flow that occurs where objects traveling in opposite directions meet.

(a) Aperture problem.



(b) Occlusion "turbulence".

Fig. 2: Common problems in optical flow estimation.

Thus, while *accurate* optical flow estimates offer valuable information about movement in the scene, it is neither complete (due to the aperture problem), nor free of severe estimation errors, in particular near occlusion boundaries.

D. Object entry and exit

Currently available datasets usually have tracked objects in the center of the first frame, however, we have initialization more challenging when objects enter the scene in a variety of ways: approach from a distance, enter from the image boundary, appear from behind an occluding object—moving or stationary, and become visible due to changes in lighting or background conditions. Termination has similar challenges—vehicles may disappear temporarily behind obstructions or due to changing conditions, they may linger near the edge of the screen, exit the scene while behind a moving vehicle, or disappear slowly into the distance. It is usually natural to terminate tracking when sequence ends or object leaves the scene in short sequences, whereas in practice, it is hard to distinguish between exiting and temporal occlusion when the object is not visible in long-time videos.

As automatic initialization and termination are missing in the majority of current tracking literature, the generally accepted methods either heuristically manually label an entry/exit area beforehand, under the assumption that objects always enter or exit within a certain area, or rely on fully manual initialization. The first method is too simplistic for general purpose vehicle tracking, and the second is impractical under constrained expense/time budget for large-scale use.

In the only available literature that explicitly addresses the effect of tracker initialization [11], the author concludes that slight temporal and spatial variation would result in performance difference. However, unlike the currently available dataset, vehicles frequently encounter significant scale change while leaving or entering the scene in the traffic surveillance videos available to us. This presents a unique problem for initialization, as early initialization would result in a small, poor-quality image, and late initialization results in missing of information due to the short trajectory. Thus, striking the right balance between tracking performance and lifetime is the key to automatic tracker initialization.

III. Proposed method

Fig. 3 describes the workflow of our automatic tracking application. We first apply background subtraction and object detector on each frame. This generates two sets of candidate boxes, which are used to initialize and update trackers. Each tracker is represented by an individual Kalman filter (see [12]). Optical flow is also computed. Any flow that matches a tracked object both by location and velocity, is used for tracker update. Tracking is terminated based on object location, velocity and time; short-lived or otherwise spurious objects are filtered out.

A. Model definition

We use Kalman filter to smooth out the noises in the observed measurements by the aforementioned components and more importantly, to integrate the strengths and compensate the weakness of each component. The *prediction* by its linear model is *corrected* with measurements observed over time, therefore the generated estimation is much smoother despite the noises in measurement input. For such a continuous system, we define a time unit dt, which is the time interval we perform an update, in our case, the time between two consecutive frames. Each variable has its own value at a certain time step t, indicated by the subscript. The prediction is performed as follows:

$$\hat{\mathbf{x}}_t^- = A\hat{\mathbf{x}}_{t-1} \tag{1}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{2}$$

Here $\hat{\mathbf{x}}_{\mathbf{t-1}}$ and $\hat{\mathbf{x}}_{\mathbf{t}}^-$ are internal states before and after prediction at time t. P_{t-1} and P_t^- are prior and post error covariances, and Q is the process noise covariance. In our case, we define the internal state a 10-dimensional vector:

$$\mathbf{x} = [x, y, w, h, x', y', w', h', x'', y''],$$

corresponding to the object's top left location (x, y), size (w, h), as well as the velocity (x', y'), rate of growth (w', h') and acceleration (x'', y''), respectively. The dynamic model A is defined based on the physics equation of displacement with velocity and acceleration. With the assumption that the object has constant acceleration within dt, we have:

$$x_t = x_{t-1} + x'_{t-1} \cdot dt + \tfrac{1}{2} \cdot x''_{t-1} \cdot dt^2 \tag{3}$$

$$y_t = y_{t-1} + y'_{t-1} \cdot dt + \tfrac{1}{2} \cdot y''_{t-1} \cdot dt^2 \tag{4}$$

$$x'_t = x'_{t-1} + x''_{t-1} \cdot dt \tag{5}$$

$$y'_t = y'_{t-1} + y''_{t-1} \cdot dt. \tag{6}$$

For width and height, we instead assume constant growth rate within dt, thus

$$w_t = w_{t-1} + w'_{t-1} \cdot dt \tag{7}$$

$$h_t = h_{t-1} + h'_{t-1} \cdot dt \tag{8}$$

After prediction, the estimated state $\hat{\mathbf{x}}_{\mathbf{t}}^-$ is corrected by an observed measurement $\mathbf{z}$ at each time step by the steps below:

$$K_t = P_t^- H^T (H P_t^- H^T + R)^{-1} \tag{9}$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + K_t(\mathbf{z}_t - H\hat{\mathbf{x}}_t^-) \tag{10}$$

$$P_t = (I - K_t H) P_t^- \tag{11}$$

In our case, we have a 10-dimensional measurement vecotr: $\mathbf{z} = [x^{bg}, y^{bg}, w^{bg}, h^{bg}, x^{det}, y^{det}, w^{det}, h^{det}, v_x, v_y]$, which represents the top left coordinates (x, y), width (w), height (h), reported by background subtraction bg and detector det, separately, as well as the velocity (v_x, v_y) reported by optical flow estimation. The measurement model H is a 10-by-10 matrix of zeros except for ones at $(0,0)$, $(1,1)$, $(2,2)$, $(3,3)$, $(0,4)$, $(1,5)$, $(2,6)$, $(3,7)$, $(4,8)$, $(5,9)$, signifying that the background and detector boxes directly measure x, y, w, and h, and that optical flow directly measures x' and y'. In other words, there are no direct measurements of w', h', x'' or y'' since they are not observable. R is the measurement noise covariance, indicating the noisiness of measurement. By manipulating the measurement noise covariance R, we indirectly adjust the Kalman gain K_t, indicating the weight of the measurement to update the corresponding prediction.

B. Measurement acquisition

None of our input measurements: background subtraction, object detection and optical flow, correspond directly to a single tracked object. Instead, they generate boxes and flow

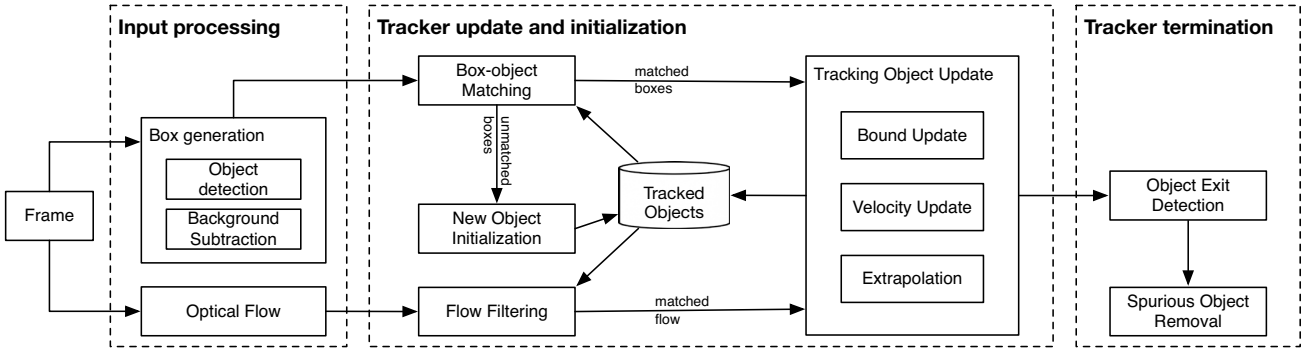


Fig. 3: Overview of proposed system. Separate Kalman filter state is initialized, maintained and terminated for each tracked object. The state is updated based on matching input from background subtraction, object detection and optical flow.

indications for an entire frame. To produce input to an individual tracked object's filter, we first compute a matching of input data to tracked objects, then apply the matched boxes and flow to the corresponding object's Kalman filter state.

Background subtraction and object detection: Both background subtraction and object detection generate bounding boxes $\{x, y, w, h\}$. We only take those consistent with tracker's current state as the measurement. For each Kalman filter, the best matching box as measurement maximizes the total overlap between the predicted bounds of internal filter, and the bounds of the measured boxes. Boxes must overlap with the predicted state in order to be considered a match. Any remaining boxes are used to initialize new tracked objects.

Optical flow: Optical flow reports velocity on a pixel-by-pixel basis, with many erroneous flow vectors due to the aperture and turbulence problems described earlier. To produce a velocity measurement from the optical flow field for a single object, we first denoise the flow field by forward-backward error thresholding, as described in [13]. We then filter the flow by location, velocity magnitude and direction. In particular, we only consider flow vectors that originate from the location of the object in the previous frame, follow the similar direction (within in 45°) of the previous state—to reduce the effect of turbulence problem, and only keep those vectors that fall within twice the error covariance (available in P, diagonal entries corresponding to the object velocity in $\hat{\mathbf{x}}$), using the simplifying assumption that flow errors follow a normal distribution. Finally we use the mean of the remaining flow vectors on horizontal and vertical direction, respectively, as the optical flow measurement for the object.

C. Model update

The measurements above are directly plugged into Eq. (10), however, are of different quality, and oftentimes some of the measurements are missing entirely. For example, the invalid foreground is generated when occlusion happens; detector misses a small-sized object, or aperture problem gives zero movement of object. We address these problems by varying the measurement noise covariance accordingly, which in turn causes the Kalman filter to choose a gain K that maximizes the quality of the internal state. Recall in Eq. (9), a large measurement covariance Q would result in a smaller K, therefore the measurement weights less in correction. When a measurement is missing, we use the value already in $\hat{\mathbf{x}}$ as a proxy, and set the covariance to ∞. Consequently, once none of the three measurements is available, the tracker merely relies on the internal prediction, which we call *extrapolation*. The tracker could still generate tracking results by the internal linear model during extrapolation. The other two cases in Fig. 3 are when at least one bounding box is available (bound update) or only optical flow is obtained (velocity update).

We also apply error gating to the background subtraction and object detection measurements. For example, only foreground bounding boxes that are well overlapped with their corresponding tracked objects (more than 30% overlap) are used for update. Meanwhile, optical flow have a lower confidence with zero value on both directions, since we have no idea whether the object is still or the aperture problem happens. In addition, as described in §II, the three measurement types naturally have different error covariances. Although detector has a higher missing rate, the precision is also high. Therefore, we assign a smaller noise covariance to measurements from the detector, than to those from the background model.

D. Tracker initialization and termination

Our system initializes new trackers based on unmatched boxes from background subtraction and object detection. This supports the challenging cases described above, but can result in many spurious trackers due to the noisy nature of both background subtraction and optical flow. We use a two-pronged approach to limit such spurious results. First, initialization is limited to objects larger than 10×10 pixels. This reduces the number of trackers in flight, without significant negative effect: cars that could reasonably be captured tend to be larger than that in our videos, except when approaching from or driving toward the vanishing point. Second, trackers are terminated when the object leaves the frame, after no direct observations (boxes or optical flow) has been made for 50 frames, in other words, in *extrapolation* mode. Any object would have such 50 frames before exit. However, upon termination, tracked objects are validated based on the number of observations and distance traveled. Spurious noise tends to be stationary and short-lived, whereas vehicles typically follow a continuous and long-lived path through the scene. To adapt to the variety of videos

and objects, distance threshold is dynamically computed by object size and video resolution. One good consequence of such scheme is that many early initialized noises are quickly discarded, since usually there is no consistent measurement available for them, while those tiny objects discarded as noises are soon available for future initialization, with better quality. Therefore, real small objects are able to be initialized at the earliest point and survive with a complete trajectory.

IV. Evaluation

To the best of our knowledge, there exists no public traffic surveillance video dataset containing complex real world interactions and illumination variations. Existing vision datasets are either not applicable to our scenario with different viewpoint (driver's view) [14] or contain short clips with limited adversarial conditions, scale changes, and illumination variations [15], [16]. Even in the largest dataset collected [16], only 15 out of 98 videos exceeds 1000 frames (33 seconds).

We collected 11 representative surveillance videos across our state, from the local department of transportation, and annotated these using VATIC [17]. Each object has its location and extent annotated on every frame, which is used as our ground truth. The average length of each video is five minutes (around 9000 frames), sufficient to cover several traffic signal cycles with real-world vehicle interactions and movement patterns. We divide the videos into two groups: simple low resolution (lowRes) and complex high resolution (highRes). Fig. 4 shows screen shots from this dataset, and Table I gives an overview of our dataset, where the rightmost four columns indicate the number of videos reflecting various challenging aspects: occlusion, shadows, distortion and pedestrians.

TABLE I: Dataset overview. The second and the third columns show the resolution and object size range in pixels, followed by number of videos under each group. The rightmost four columns show the number of videos reflecting various challenging aspects (occlusion, shadow, distortion and pedestrian).

Group	Resolution	Object size	#	Occ.	Shad.	Dist.	Ped.
lowRes	342 × 228	32–44,814	5	3	1	3	0
	320 × 240	48–25,284	2	2	1	2	0
highRes	720 × 576	84–255,106	4	3	0	0	1

A. Tracker configuration

To better evaluate how each component contributes to the tracker in videos under various conditions, we use three different configurations of our tracking system, based on the inputs used. BG uses background subtraction and optical flow, DET uses the object detector and optical flow, and BG+DET uses all three inputs. We also run several state-of-the-art trackers, including KCF [1], STRUCK [3], ASMS [2], DAT[4], staple [18], MEEM [19] and SRDCF [20] as baselines. Since all these trackers require manual initialization (namely human input), they are not able to be compared with our tracker directly. Instead, we can only partially evaluate them by providing manual initialization, see §IV-D. By carefully selected manual initialization, the comparison here is in favor of the baselines.

B. Evaluation metrics

Given a set of generated and ground truth object trajectories, usually represented by a sequence of rectangular bounding boxes, we desire one or more performance metrics that capture the accuracy of the proposed system. Aspects that need to be captured include 1) tracking duration—if a ground truth trajectory of an object contains N frames, for how many of these frames does the system track the object, 2) recall—how many of the ground truth trajectories were represented in the tracking result, 3) precision—what proportion of tracking results had a corresponding trajectory, as well as 4) the overlap between the tracking result and the ground-truth.

Note that our problem does not fit into common multi-object tracking setting, since moving objects are tracked individually, instead of being modeled globally. Common metrics such as MOTA [21] would generate a meaningless number since we could have potentially more objects tracked than ground truth.

As the first step, we match each ground truth to a tracking result, by maximizing the accumulated overlap ratio r,

$$r = \frac{\sum_t (S_t^{gt} \cap S_t^{traj})}{\sum_t (S_t^{gt} \cup S_t^{traj})}, \tag{12}$$

where S_t^{gt} and S_t^{traj} are ground truth and trajectory box areas at frame t, respectively. Note that the sum is computed over the union of trajectory and ground truth lifetime. Eq. (12) ensures the ground truth is matched to a longer trajectory with reasonable coverage. Additionally, to filter some spurious objects slightly touched the ground truth, we also require any match to have more than 30% overlap on at least one frame, where the value is set experimentally.

C. Object-level evaluation

Given the matched results, we can calculate the proportion of ground truth trajectories that were matched to the tracking result (recall), and the proportion of tracking results that had a matching trajectory (precision), for our three trackers and two types of videos. Figs 5–6 show the results in absolute numbers, also to provide some perspectives on the scale of the evaluation. The true positives are those correctly tracked objects and false positives are noisy objects. Our best tracker configuration, BG+DET, achieves 81% recall and 87% precision on our low resolution, simple videos, and 65% recall, 57% precision on the high-resolution, complex scenes. Here, the majority of failures in the complex scenes were due to complicated interactions and heavy occlusion throughout the scene, where vehicles were only partially visible. Currently, we do not split such merged objects into their constituent parts; we leave this for future work.

Comparing the three configurations, we see that the detector performs quite poorly on the low-resolution scenes, due to small and poor quality imagery where it is sometimes difficult even for a person to correctly identify an object as a vehicle. By contrast, background subtraction does quite well on these uncomplicated scenes. For the higher resolution, more complex scenes, the detector performs well, while the background subtraction model struggles and largely introduces noise.

Fig. 4: Snapshots of videos in our dataset, with various resolution, viewpoint, illumination, vehicle size and interactions. In particular, (c) shows shadows; (f) and (g) show severe distortion by fish-eye camera. We group these videos by their characteristics: (a) - (g) are simple low resolution videos (lowRes), and (h) - (l) are complex high resolution videos (highRes).

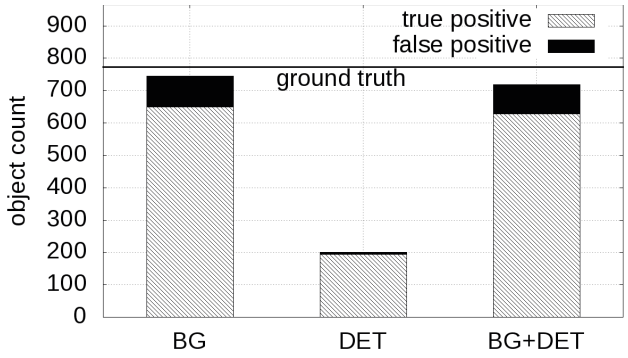


Fig. 5: Tracked object count under different tracker settings on low resolution, less complex scenes.

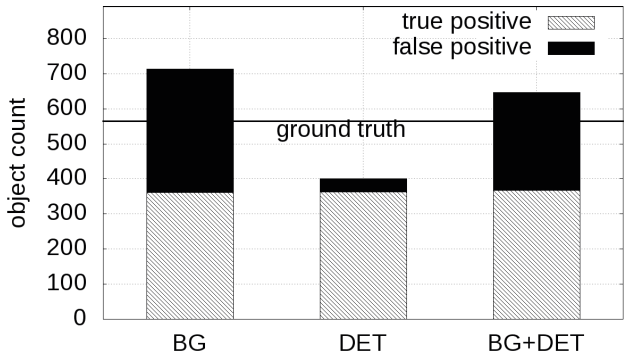


Fig. 6: Tracked object count under different tracker settings on high resolution, more complex scenes.

D. Pixel-level evaluation during entire lifetime

We are also interested in evaluating how well the tracker follows objects in the scene. For example, as one of our motivating application, accurate tracking is necessary for correct turning movement classification, as Fig. 4(a) illustrates. While there exists no similar work or benchmark that incorporates automatic initialization, we can only partially compare our system to prior work. Here, we modify our tracker to accept manual initialization. Both our modified trackers and the baseline trackers are initialized with boxes extracted from ground truth. These "initialized" trackers are then evaluated along with our automatically initialized trackers. Note that the experiment is designed in favor of those manually initialized trackers, since the initialization boxes are from ground truth and perfectly match the object in the scene.

To accept manual initialization in scenes with significant scale change, each tracker is initialized with the first box of ground truth larger than a threshold:

$$S_{thresh} = \left[\max(S^{gt}) - \min(S^{gt})\right] * s + \min(S^{gt}), \quad (13)$$

where $\max(S^{gt})$ and $\min(S^{gt})$ are the maximal and minimal ground truth areas along the sequence. $s \in \{0, 0.2, \ldots, 1.0\}$, corresponds to the minimum fraction of the maximum object size at which initialization may occur. Note s only affects objects that enter with an increasing size; while shrinking objects would be initialized upon appearance regardless of the threshold. Tracking is terminated at the last frame of ground truth. We arrived at this design as some trackers tend to perform poorly when initialized by small images, yet delaying initialization until the object grows large can result in arbitrarily shortened trajectory.

Figs 7–8 compare the overlap ratio of our three tracker configurations both with manual and automatic initialization (horizontal lines), as well as the overlap ratio of several other trackers on our high- and low-resolution datasets. The x-axis is the initialization threshold s in Eq. (13), and the y-axis is the overlap ratio r from Eq. (12), averaged over all objects. This is computed from the first to the last frame of ground truth for the manually initialized and terminated trackers. While for our automatic trackers, it is computed over the union of tracking and ground truth lifetimes. This accounts for both tracking accuracy and lifetime, as we illustrated before.

The shape of the curves for the initialized trackers captures the trade-off between early-and-inaccurate, vs. late-but-accurate initialization. Overall, our BG-based trackers handily outperform other trackers when manual initialization and termination is provided. More importantly, the auto-initialized BG trackers essentially meet the performance of other trackers on the simple videos, and significantly outperform them on the complex videos. We hypothesize that the automatic initialization allows the tracker to better adapt to individual vehicles vs. the constant threshold set for manual initialization.

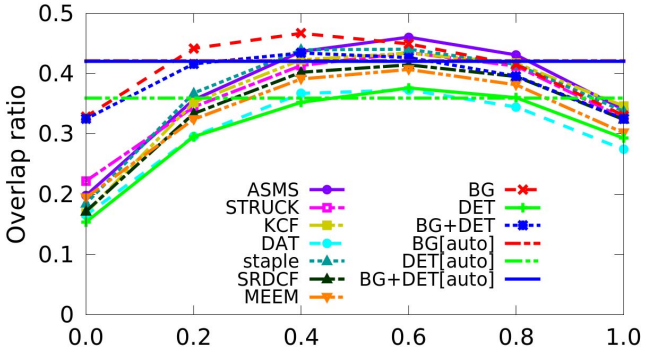


Fig. 7: Overall overlap ratio on low resolution videos.

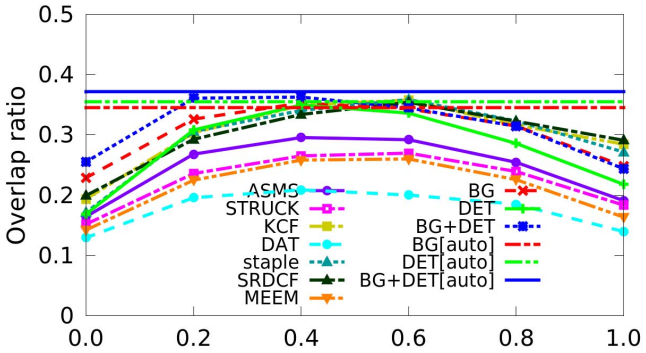


Fig. 8: Overall overlap ratio on high resolution videos.

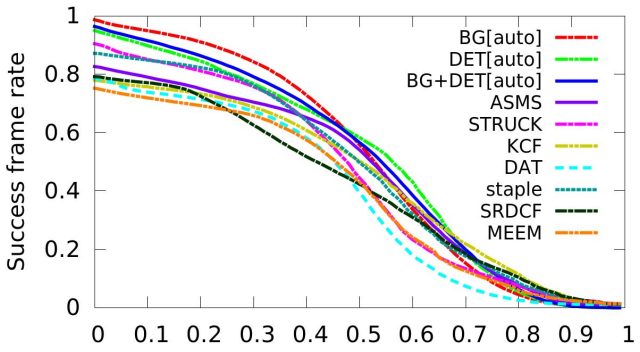


Fig. 9: Success plot on low resolution videos.

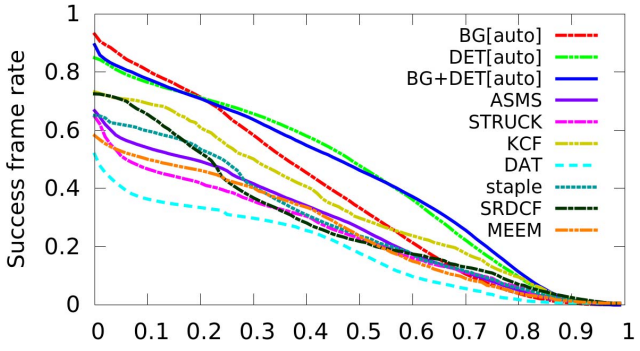


Fig. 10: Success plot on high resolution videos.

Additionally, consistent with our conclusion in §IV-C, detector dominates the tracking update on high resolution videos, while background subtraction plays its role on low resolution videos.

E. Pixel-level evaluation during tracked period

Then we focus on the performance during only the tracked period. Borrowed from standard single object tracking measurement, we show the success plot for trackers for two video groups, in Fig. 9 and 10 respectively. Different from above, we compute the success frame rate from the initialization frame, instead of entire ground truth sequence. The performance is measured by the area under the curve. By Figs 7-8 we see overlap ratio between 0.2 and 0.6 is a good balance of tracking accuracy and trajectory completeness, we hereby show results of initialization threshold of 0.4 here. Our automatic trackers significantly outperform other trackers with manual initialization. This demonstrates that proper initialization is critical to the tracking performance and similar conclusion about the contribution of each tracking component can be drawn.

F. Throughput

Finally, we explore the throughput of trackers on different resolution videos, as shown in Fig. 11. The evaluation was done on a Linux desktop with an Intel Core i7-3770 3.40GHz processor and a GeForce GTX TITAN X GPU. According to these results, BG significantly outperforms 6 out of 7 baseline trackers, with around 5× throughput improvement compared with real time (about 30 fps). Interestingly, ASMS outperforms our BG tracker on low resolution videos, however, it becomes 17× slower once it is applied on high resolution videos. Although the use of the object detector (DET, BG+DET) makes the tracker slower, they are the only ones that maintain a similar frame rate on both low- and high resolution videos near real time, showing good scalability of resolution.

V. Related Work

Object tracking: Object tracking algorithms fall into either multi-object or single object tracking category. Multi-object trackers deal primarily with the combinatorial task of associating sets of detections across frames, often modeled as a graph-based global optimization problem [22], [23]. Such methods are computationally expensive, and typically impractical in time-critical applications. Meanwhile, single object tracking has experienced a rapid progress with the help of robust feature representation [24] and better learning scheme such as SVM [3] and boosting [25]. However, object tracking still remains a challenging problem due to the difficulty caused by

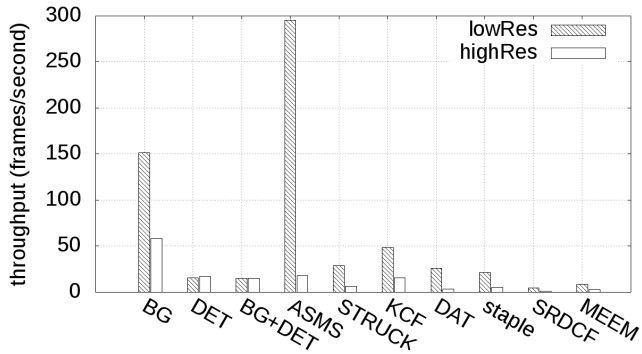


Fig. 11: Tracker throughput on different resolution videos.

changing appearance and occlusions. To cope with appearance change, better affinity measurement [26] and adaptive learning schemes [27] may help. For occluded objects, multi-object trackers tend to inherently model occlusion, for example, using augmented graph representation [22] while single object trackers maintain the memory of the object appearance [27].

Tracker initialization and termination: Trackers today are evaluated in an idealized setting, where manual initialization and termination is provided. Most multi-object trackers assume excellent detector performance, but do handle object entry/exit. They either model it intrinsically by adding entry/exit nodes to the optimization graph [28], or manually by defining an entry and exit area [29]. In [11], the only available literature we found related to tracker initialization, the authors evaluate single-object tracker initialization spatially and temporally, and conclude that spatial and temporal variation of initialization would affect the tracking performance. However, no conclusion on how to make proper initialization is made.

Vision in traffic surveillance: Recently computer vision techniques are extensively applied in traffic analysis with the wide deployment of surveillance camera, such as real-time vehicle tracking [30], vehicle counting [31], parking occupancy detection [32], anomalous event detection [33]. Compared with core computer vision algorithms, such systems face more real-world challenges and restrictions. Therefore manual input is often added to achieve reasonable performance. For instance, image–real world coordinates mapping beforehand [30], lane width on image [34], and entry region for vehicle detection [34]. Another observation is that those systems are only applicable to videos of a certain view. For example, [30] uses top-front view of high way videos, making vehicles roughly the same size. Therefore, the portability is significantly limited by manual input and task-specific applications.

VI. Conclusion

In summary, we build an efficient system aiming at challenges in real world tracking application. The system integrates the fallible background subtraction model, object detector and optical flow into an efficient automatic tracker. We also contribute to the community by releasing the first surveillance dataset containing sufficient real-world challenges. By comparative evaluation of manually initialize trackers and our automatic trackers, we validate our statement that a good tracker in real world use must balance the tracking accuracy and object lifetime. We also demonstrate robust tracking performance combined with real-time processing throughput.

References

[1] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. on Patt. Analysis and Mach. Int.*, vol. 37, no. 3, pp. 583–596, 2015.

[2] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Patt.Rec. Letters*, vol. 49, pp. 250–258, 2014.

[3] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *ICCV*. IEEE, 2011, pp. 263–270.

[4] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *IEEE CVPR*, 2015, pp. 2113–2120.

[5] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*. IEEE, 2010.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE CVPR*, 2014, pp. 580–587.

[7] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *Image Processing, IEEE Trans. on*, vol. 20, no. 6, pp. 1709–1724, 2011.

[8] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Patt.Rec. letters*, vol. 27, no. 7, pp. 773–780, 2006.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Adv. in neural info. processing systems*, 2015, pp. 91–99.

[10] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, 1981.

[11] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE CVPR*, June 2013.

[12] M. S. Grewal, *Kalman filtering*. Springer, 2011.

[13] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Patt.Rec. (ICPR), 2010 20th Int. Conf. on*. IEEE, 2010.

[14] S. Sivaraman and M. M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, pp. 267–276, 2010.

[15] S. Manen, J. Kwon, M. Guillaumin, and L. Van Gool, "Appearances can be deceiving: Learning visual tracking from few trajectory annotations," in *European Conf. on Comp.Vis.* Springer, 2014, pp. 157–172.

[16] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. on Patt. Anal. and Mach. Int.*, vol. 37, no. 9, 2015.

[17] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *Int. J. of Comp.Vis.*, pp. 1–21, 2013.

[18] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *IEEE CVPR*, 2016, pp. 1401–1409.

[19] J. Zhang, S. Ma, and S. Sclaroff, "Meem: robust tracking via multiple experts using entropy minimization," in *European Conf. on Comp.Vis.* Springer, 2014, pp. 188–203.

[20] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *IEEE Int. Conf. on Comp.Vis.*, 2015, pp. 4310–4318.

[21] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, pp. 1–10, 2008.

[22] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *IEEE CVPR.*, 2013.

[23] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE trans. on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1806–1819, 2011.

[24] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *CVPR*. IEEE, 2010, pp. 1269–1276.

[25] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting." in *BMVC*, vol. 1, no. 5, 2006, p. 6.

[26] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *IEEE ICCV*, 2015.

[27] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE trans. on patt. anal. and mach. int.*, vol. 34, no. 7, 2012.

[28] L. Zhang, Y. Li, and R. Nevatia, "Global data assoc. for multi-object tracking using netw. flows," in *CVPR'08*. IEEE, 2008, pp. 1–8.

[29] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *CVPR*. IEEE, 2011.

[30] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time comp.vis. system for vehicle tracking and traffic surveillance," *Transp. Res. Part C: Emerging Tech*, vol. 6, no. 4, pp. 271–288, 1998.

[31] W. Wang, T. Gee, J. Price, and H. Qi, "Real time multi-vehicle tracking and counting at intersections from a fisheye camera," in *WACV*, 2015.

[32] O. Bulan, R. P. Loce, W. Wu, Y. Wang, E. A. Bernal, and Z. Fan, "Video-based real-time on-street parking occupancy detection system," *J. of Electronic Imaging*, vol. 22, no. 4, pp. 041 109–041 109, 2013.

[33] F. Jiang, J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos, "Anomalous video event detection using spatiotemporal context," *Comp.Vis. and Image Understanding*, vol. 115, no. 3, pp. 323–333, 2011.

[34] Y.-L. Chen, B.-F. Wu, H.-Y. Huang, and C.-J. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," *IEEE Trans. on Indust. Elec.*, vol. 58, no. 5, pp. 2030–2044, 2011.