Real Time Object Tracking based on Dynamic Feature Grouping with Background Subtraction

ZuWhan Kim
California PATH, University of California at Berkeley, CA, USA
http://path.berkeley.edu/~zuwhan

Abstract

Object detection and tracking has various application areas including intelligent transportation systems. We introduce an object detection and tracking approach that combines the background subtraction algorithm and the feature tracking and grouping algorithm. We first present an augmented background subtraction algorithm which uses a low-level feature tracking as a cue. The resulting background subtraction cues are used to improve the feature detection and grouping result. We then present a dynamic multi-level feature grouping approach that can be used in real time applications and also provides high-quality trajectories. Experimental results from video clips of a challenging transportation application are presented.

1. Introduction and Previous Work

Object detection and tracking is a major research area in computer vision. One of its application areas is traffic scene analysis. Cameras are less costly and easier to install than most other sensors, so many are already installed on the roadside, particularly at intersections. Resultant video images are used to estimate traffic flows, to detect vehicles and pedestrians for signal timing, and to track vehicles and pedestrians for safety applications.

For decades various vehicle and pedestrian detection and tracking algorithms have been introduced, [15], [19], [16], [3], [6], [9], [14], [1], [18], and there are many commercial systems to detect vehicles (e.g., "virtual loop detectors") and pedestrians. Most of the above systems (and also many of other object tracking applications) are based on the *background subtraction* algorithm. It first extracts a static background hypothesis from a sequence of images, then calculates a difference between the background hypothesis and the current image to find foreground objects.

The background subtraction algorithm requires a relatively small computation time and shows robust detection in good illumination conditions. However, it suffers from the problems such as occlusions, the presence of shadows, and sudden illumination changes. Many efforts have been made to solve the occlusion problem, for example by applying a Markov Random Field model [9], but the result has not been satisfactory with significant occlusions, as it is an extremely difficult problem to segment occluded objects by considering only the background subtraction result.

Furthermore, it is difficult to deal with problems such as sudden illumination changes and stopped vehicles. For example, vehicles moving slowly in traffic congestions or stopped vehicles at an intersection will eventually be recognized as background objects. In addition, the trajectories are usually determined by linking the center of the object blobs, and oftentimes it results in zigzaggy trajectories.

Another approach is based on feature tracking and grouping, [3], [1]. This is done by extracting and tracking individual *corner features*, [7], and grouping them based on their trajectories. The grouping algorithms are applied to full trajectories of the corner features for post-processing applications. Since this method uses a set of long trajectories, segmentation among occluded objects is easier to perform than with background subtraction.

A challenge in applying this algorithm is that it is not always easy to robustly track a corner feature over a long period of time, especially when vehicles turn at intersections or are occluded by other vehicles or pedestrians. In addition, keeping and processing a set of long corner trajectories can be burdensome. For example, when a vehicle is waiting at an intersection for over a minute the corner feature trajectories must be kept for more than 1000 frames. Therefore, these algorithms cannot be applied to real time detection and tracking applications, especially for intersections. Another challenge is that the grouping algorithms often group nearby vehicles moving together or separate a large vehicle into two because corner features are not evenly distributed over the vehicles in many cases.

Finally, there is an approach based on appearance-based vehicle detection [15], [19], [14]. Kim and Malik [14] introduced a model-based vehicle detection algorithm eventually adopted by the NGSIM (Next Generation SIMulation) traffic modeling effort to generate a large number of long trajectories [13]. However, such an approach only works on a certain classes of vehicles (e.g., passenger cars and trucks for [13]) from a limited range of viewpoints (e.g., a bird-eye view for [13]). In addition, the tracking of [13] based on template matching does not work for intersection videos where perspective changes of vehicles are significant.

There is also active research on appearance-based pedestrian detection, for example [18]. However, many of the reliable detection algorithms require heavy computation and the robust tracking is still a problem. The algorithm we will introduce can effectively combine non-motion-based object detection algorithms or interactive detection results to get higher quality of results and to reduce computation.

We propose an approach to combine the background subtraction and the feature tracking and grouping algorithms. Our algorithm is distinguished from the previous ones that

- it gives more robust background subtraction result by using a feature tracking result as an additional cue;
- at the same time, it gives a better feature detection and grouping result by using a background subtraction cue;
- it provides a multi-level feature grouping algorithm to deal with various sizes of objects, such as for detecting passenger vehicles and bicycles at the same time; and
- it introduces a dynamic feature grouping which can be applied to real time applications and produce a high quality of object trajectories from fragmented feature tracks (as opposed to previous work's using nice long trajectories).

An approach of combining background subtraction and feature tracking is also found in [10], where Kanhere et al. used the background subtraction result to estimate the 3D heights of corner features for a low-angle camera by assuming that the bottom of the background subtraction region is the bottom of the object. Although such an assumption often fails due to occlusions, good results were reported from a number of challenging highway video clips because of redundant estimation over multiple frames. However, some of the challenges of the background subtraction and the feature tracking and grouping algorithms, such as long corner trajectories at an intersection and the illumination changes, still remain. In addition, the trajectory generation, accomplished by matching frame-by-frame detections, does not deliver high quality trajectories. The suggested height estimation algorithm may be incorporated in our framework as an additional cue to further improve the result.

In Section 2, we introduce our augmentations made on the background subtraction algorithm. The feature grouping algorithm is introduced in Section 3. The result on various applications are presented in Section 4, and the conclusion and future work is presented in Section 5.

2. Background Subtraction

A typical background subtraction algorithm applies a Kalman filter (or α-blending) to the pixel intensities to find the background [11]:

$$B_{t+1} = B_t + (\alpha_1(1 - M_t) + \alpha_2 M_t)D_t, \tag{1}$$

where B_t represents the background model at time t, D_t is the difference between the present frame and B_t, and M_t is a binary moving object hypothesis mask.

Such an approach works well when foreground objects appear infrequently, but when the background is occluded by an object for a significant time, the algorithm begins to fail. Another problem is that M_t is usually generated from D_t by thresholding and applying morphological operators. Such self-feedback can make the filtering unstable. For example, a single detection failure or a sudden illumination change can result in a permanent failure (or a *ghost*) which may even grow in size until it covers up the entire image. Sudden illumination changes commonly occur in many field video images because most video cameras have an auto-iris feature.

Various augmentation has been applied to the background subtraction, for example, to use temporal median instead of the α-blending [4]. More recently, Batista et al. introduced various augmentations including the use of multi-layer background models and dynamic thresholding [2]. Such augmentations significantly improve the robustness, but the problem of self-feedback is still there.

Therefore, we incorporate an external cue (corner features) to generate more robust M_t. In addition, we also made the following modifications to Equation 1:

- the temporal median approach is combined with the α-blending;
- an illumination correction procedure is added to deal with sudden/temporary illumination changes; and

We use an update equation

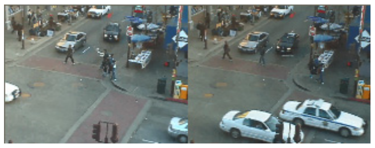
$$B_{t+1} = \begin{cases} Ic(B_t) & M_t = 1 \\ Ic((1-\alpha)B_t + \alpha N_t) & M_t = 0, \end{cases} \tag{2}$$

where $Ic()$ is an illumination-correction function and N_t is the temporal median of the recent, say 15, frames. Note that our background update rate is about 2 frames per second and the 15 frames spans about 7 to 8 seconds.

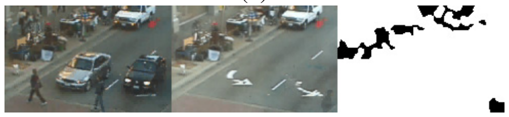
The illumination-correction is applied to each of the RGB value since the auto-iris can also change the color distribution (hue):

$$Ic(R, G, B) = (k_R R, k_G G, k_B B), \tag{3}$$

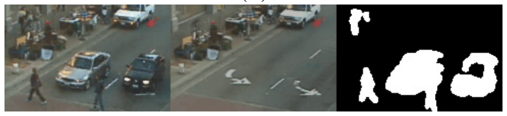
where k_R, k_G and k_B are determined by voting on R_C/R, G_C/G, and B_C/B over all the pixels in the images. (R_C, G_C, B_C) are the pixel values of the current frame.



(a)



(b)



(c)

Figure 1. (a) The entire scene suddenly becomes dark by an auto-iris camera as the two white vehicles in the bottom pass by. (b) A disastrous detection result without the illumination correction. (c) An enhanced result with the illumination correction.

For M_t, we start with the standard procedure which is to 1) threshold the difference, 2) apply morphological operators (or threshold after over-smoothing), and 3) perform connected component analysis to fill holes, remove small regions, and find object blobs. After the object blobs are found, we apply an additional validation step to remove the ghosts. We assume that within all the non-ghost foreground region there exists at least one *valid corner*, i.e., a corner feature which is not found from the background image. For more details on the valid corner, see Section 3.

An example result is shown in Figure 1 where the illumination challenge caused by an auto-iris camera. The two white vehicles in the bottom changes the entire scene darker (Figure 1a) and it causes a significant false alarms (Figure 1b). However, the error is minimized by applying the illumination correction.

The resulting object blobs are not the final result but they are used as supplementary cues in the feature tracking and grouping which will be discussed in the next section.

3. Feature Tracking and Grouping

Previous feature grouping work, [3], [1], groups corner features directly into objects using proximity and motion history. Such a single-level grouping is difficult and/or computationally expensive, especially when we deal with objects of different scales (for example, bicycles, passenger cars, and trucks). For instance, the distance between two corner features that belong to the same vehicle can be much larger than the two corner features that belong to two nearby vehicles, which can confuse the grouping algorithm. However, when we apply a sophisticated grouping algorithm to handle this it will bring in computational burden particularly when comparing long trajectories of corner features.

To efficiently deal with the problem, we present a multi-level grouping where individual corner features are first grouped into small clusters (“cluster features”) then the cluster features are grouped into object-level features.

The grouping is performed on each frame (*dynamic grouping*) as opposed to the previous efforts, [3], [1], which applied the grouping algorithms on the final tracking results. Therefore, the proposed algorithm can be applied in real time. Note that one of our main goals is to generate a trajectory of an object. Therefore, directly applying conventional grouping algorithms, such as K-means and the Normalized-Cut [17], frame-by-frame basis will not provide solid trajectories.

3.1. Corner Feature Tracking

The lowest-level corner features are detected by finding the eigenvalues of the local sums of derivatives [7]. The corner features are only detected in the *foreground region* which is estimated by the background subtraction algorithm. The detected corners are tracked by applying cross-correlation template matching on a small image patch (9×9 in our implementation). The search window sizes for the match vary from the applications but we first apply a large window (for example, 15×15) when the direction/speed of the corner is not known, then a small window (for example, 7×7) near the expected position estimated by the previous movement.

The tracked corners are validated with comparison of the background image: another template matching search is performed on the background image with a small search window size (3×3 in our implementation). When a corner has a match in the background image it is considered invalid and removed. Such invalid corners are often generated by errors by tracking failures such as drifting or errors in estimating the foreground region.

We consider a corner feature ‘valid’ (see Section 2) when it is tracked over a number of (three in our implementation) consecutive frames, does not have a match in the background image, and is neither moving or picked-up by an existing cluster (see Section 3.2). When a corner feature matching fails over a number of (five in our implementation) consecutive frames it is no longer used. Corner features are detected in each and every frame, and those not overlapping with existing ones are subject to tracking.

Note that a feature trajectory can comprise several thousand frames long in traffic video images due to a long signal waiting times at signalized intersections. However, we do not need to keep the whole thousand frames of corner trajectories but just for, say, 20 recent frames in our implementation because of our dynamic multi-level grouping approach. The resulting corner trajectories (the *corner tracks*) are shown in Figure 2. We see that some of the corners were tracked over many frames while some were not.

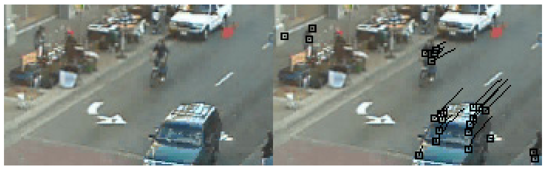


Figure 2. Corner tracks. Individual corner feature tracking fails due to various reasons. This results in fragmented trajectories.

3.2. Cluster Tracking

The next step is to group the corner tracks into small clusters, with each represented by a circle. An ellipse is also used for an application with single-level grouping (Section 4.1). We apply a variation of an Expectation-Maximization (EM) algorithm [5] for the grouping. For each cluster, its expected position and size in the new frame are determined from its current "member" corner features. Then, for each corner feature, its membership in the current frame is re-determined by comparing its position, history of motion, past trajectory, previous cluster membership, and background-subtraction-blob membership history to their "parent cluster". The position and the size of the cluster are re-determined by applying the updated membership. Then after a small number of iterations (3 in our implementation), the final position and the size of the cluster is determined.

To increase the robustness we apply α-blending to determine the position and the size of a cluster. First, the current position and the size of a cluster is estimated by finding a robust mean (for position) and a standard deviation (for size) of the positions of a good majority, say 90%, of the corner tracks that are close to the initially estimated position. Then the actual size of the cluster is determined by applying α-blending. For example,

$$r_t = (1-\alpha)r_{t-1} + \alpha r_c, \tag{4}$$

where r_t is the radius of the current cluster, r_{t-1} is for the previous frame, r_c is the radius estimate from the current frame, and α is a parameter that gradually drops down as time goes by. For example, we use $\alpha = \lambda/(n+\lambda)$ where n is the length of the trajectory and λ is a constant drop ratio ($\alpha = 1$ when $n = 0$).

A Bayesian reasoning was applied to determine the membership of a corner feature. Given a cluster, a corner feature's membership is given as:

$$P(\text{member}|p,r,m,t,c) = \alpha P(p,r,m,t,c|\text{member}), \tag{5}$$

where $\alpha = \frac{1}{P(p,r,m,t,c|\text{member})+P(p,r,m,t,c|\neg\text{member})}$ is the normalization parameter, p is the proximity between the cluster and the feature (relative distance from the ellipse boundary), r is the background-subtraction region history (ratio of being in the same blob), m is the motion history (ratio of incompatible motion), t is the trajectory compatibility (maximum disparity of the two trajectories), and c is whether the previous cluster membership. We assume conditional independence of individual evidence variables to estimate $P(p,r,m,t,c|\text{member})$.

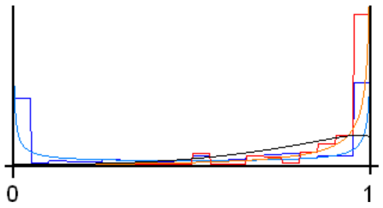


Figure 3. An example pair of probability distributions used for the grouping algorithm: the background subtraction region. The red and blue lines are histograms for the positive and negative data, and the orange and cyan curves are the fitted distributions (beta distributions). The black curve shows the likelihood of the fitted distributions.

The individual probability distributions were estimated by using a semi-supervised learning procedure. First the algorithm was ran with manually-assigned parameters to obtain a reasonable result. In addition, a user interface was developed to correct observably erroneous membership assignments. Thus we can obtain a reasonably good grouping dataset with a few manual intervention. Finally, parametric probability distributions was fit to the collected dataset. We used half-Gaussian, gamma, beta, and step (for filtering) distributions to approximate the individual distributions. For example, the positive and negative background-subtraction region history were fit to a pair of beta distributions as shown in Figure 3. Note that the cluster grouping is a hidden-level with no ground truth. Therefore, the proximity parameter was manually modified such that each cluster maintains a reasonable size.

Most previous work uses frame-by-frame position estimates to generate trajectories. However, such estimates do not give good quality of object trajectories because 1) corner features appear and disappear and/or 2) and we do not get identical grouping all the time. For example, [3] used the center of mass in each frame to generate a trajectory, which became too noisy. The background-subtraction-based tracking also gives such noisy trajectories because the center of a blob is not always the center of the object. However, transportation applications, such as analysis of driver behaviors, require a fine quality of trajectories, and such a noise cannot be accepted. Also, estimating a good cluster trajectory helps the grouping process.

To get a fine level of trajectory, we keep the position of the cluster (α-blended center position) and the velocity history of the cluster separately. The velocity of the cluster is set to a robust average of the velocity of the member corner features. The resulting trajectory is estimated based on the accumulated velocity but a possible drift is adjusted by using the current and previous positions of the cluster. More details on the trajectory estimation are presented in

Section 4.3.

A new cluster is generated from corner features which do not belong to any of the existing clusters (*emerging corner features*). The emerging corner features are clustered based on their positions and the background-subtraction blob memberships. A Normalized-Cut [17] was applied for the clustering – each feature is represented as a node on a graph and the edge weights $w(u, v)$ between nodes u and v are set to the distances between the two features when they are in the same blob and have similar trajectories or 0 otherwise. Then a graph cut is found to minimize

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}, \qquad (6)$$

where A and B are disjoint partitioning of the graph V, $cut(A,B) = \sum_{u\in A, v\in B} w(u,v)$, and $assoc(A,V) = \sum_{u\in A, t\in V} w(u,t)$. The solution is found by computing the eigenvectors of the smallest eigenvalues from the edge weight matrix.

3.3. Object-level Tracking

Finally, the clusters are grouped into objects. The same EM-style algorithm for the corner feature grouping is used for grouping the clusters but with a few differences:

- the shape of an object is fit to an ellipse instead of a circle;
- the position and the shape of the final object is determined not by the positions of the member clusters but by the positions of the member corner features of the member clusters; and
- 3D criteria is used in determining the minimum and maximum sizes of the ellipse. In addition, the trajectories are compared in 3D space (assuming a fixed height, say 0.5 meter in our implementation) to determine the membership.

It is necessary to group clusters based on their 3D trajectories because the disparity among the trajectories can be quite large in the image space (in pixels) due to camera perspective.

An example cluster and object grouping result is shown in Figure 4. Nice and long trajectories are obtained from the fragmented corner tracks (Figure 2).

4. Experimental Result and Applications

We present a result on an intersection video clip of about 80s with vehicles, bicycles, and pedestrians. This video clip is particularly challenging because of occlusions among all road users in addition to having a cluttered background. It also contains some challenging illumination cases as presented in Figure 1. Example results are shown in Figure 5. The video clips contains

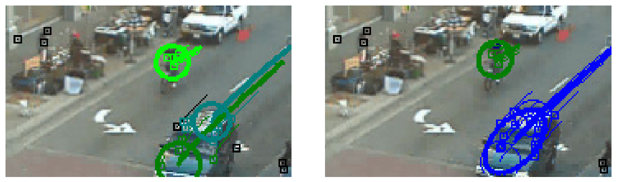

(a) (b)

Figure 4. Nice and long trajectories are obtained from fragmented corner tracks. Shown are: (a) cluster grouping result, and (b) object-level grouping result.

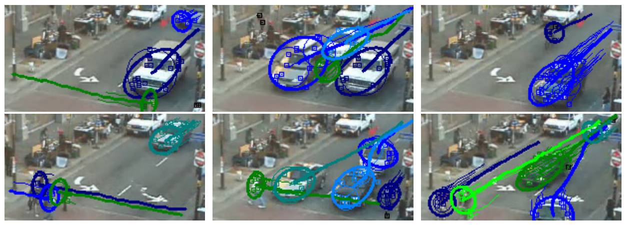


Figure 5. Vehicle and bicycle detection and tracking results.

total nine vehicles, seven bicycles, and many pedestrians on the road (including a crossing). Our algorithm correctly detected the positions, sizes, and trajectories of all these road users/objects. Most of the pedestrians and bicycles in the crossing and the cluttered background were also detected by the algorithm with a small number of under-segmentation errors and missed detections. The original and processed video clips can be downloaded at `http://path.berkeley.edu/~zuwhan/ztracker`.

The average processing time was 45ms on a 1.8GHz Pentium Core processor. The algorithm was applied at 10 frames per second. The processing time rather depends on the scene complexity than the size of the image as the corner tracking takes a significant portion of the total computation.

The result was compared with the frame-by-frame application of the NCut trajectory segmentation (cf. Section 3.2). The background subtraction result was not applied but all other learned probability parameters (except the proximity where corner *vs.* corner is different from corner *vs.* cluster) were applied identically. It showed a reasonable segmentation performance for many objects in many frames, but detection and grouping errors were observed in about a half of the frames. Some under- and over-segmentation results and noise picked-up by the NCut grouping algorithm were shown in Figure 6. In addition, the algorithm requires keeping trajectories of corner features for as much as 1000 frames. Therefore, it required a significant processing time to compare the long trajectories, and the average processing time was 77ms. On the contrary, our dynamic feature grouping requires keeping only 20 frame-long trajectories for corners and 1000 frame-long trajectories were only needed for a small number of clusters. Considering that the 45ms processing time also includes the processing

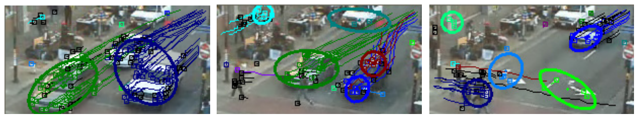


Figure 6. Under- and over- segmentation errors and false detections when the Normalized Cut was directly applied frame-by-frame without background subtraction.

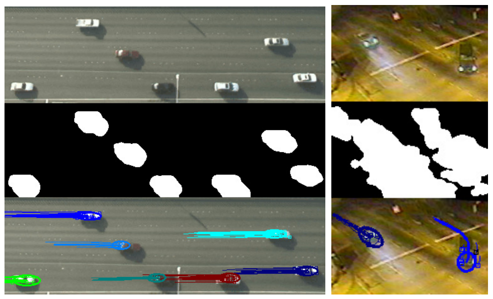


Figure 7. Results on challenging illumination conditions where the background subtraction algorithms fail to correctly localize the vehicle due to the shadows and the headlights. The proposed algorithm shows much better localization performance.

time for the background subtraction, the benefit of dynamic grouping is significant.

Additional results on other challenging video images are shown in Figure 7. We see that the algorithm is more robust to typical background subtraction challenges of shadows and headlights. Since these causes only a small number of robust corner features, our robust ellipse fitting delivers a good localization performance.

Our algorithm can be applied to various object detection and tracking applications. We next present its applications on pedestrian detection and tracking, bicycle detection and classification, and interactive trajectory extraction.

4.1. Pedestrian Detection and Tracking

Detecting, counting and/or tracking pedestrians can be used for various safety and traffic engineering applications. Since the shape and the size of pedestrians are roughly uniform, we use a single-level grouping with a fixed cluster shape (a vertically ellipse). An example detection and tracking result from a pedestrian crossing video clip is shown in Figure 8. The video clip is challenging, with clusters of pedestrians move together. Therefore, the resulting background subtraction blobs are in clusters as shown in the image. The video clip contained total 17 pedestrians crossing the road, and there was only one false alarm and one grouping error (shown in the left) where three people walking together were initially detected as two (which was subsequently corrected). There was also one tracking failure, but the person was also re-detected later. The resulting video clip can also be downloaded at the same website.

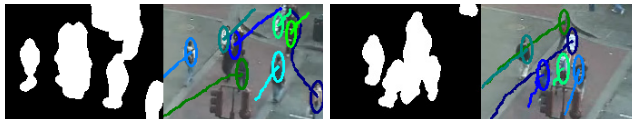


Figure 8. An application to pedestrian detection and tracking. A good result was obtained on a very challenging video clip of a pedestrian crossing.

4.2. Bicycle Detection and Classification

Bicycles represent an important transportation modal alternative, but many conventional detectors, including the typical inductive loop detectors, fail to detect bicycles. Bicycles are significantly smaller in size than passenger vehicles, so detecting both bicycles and passenger vehicles is challenging. Our multi-level grouping algorithm enables simultaneous detection of vehicles and bicycles. In addition, we can use the result to classify bicycles from other vehicles, and for example, to provide an extended signal for bicyclists to cross intersections.

Since we know the position and the size of the object, we can apply a simple binary classifier to distinguish non-vehicles (bicycles and pedestrians) from vehicles. We applied a support machine vector classifier suggested in [12]. Given a fixed size (in world coordinates) image window, we extracted a corner density, horizontal line density, vertical line density, and the standard deviation of pixel intensities of the region. Then a support vector machine was applied to train the classifier.

An example result is shown in Figure 9. The video clip is very challenging because the shadow casts of a tree moves due to wind and a sun angle change (as the shadow position of the left-most image is different than in the right-most image). The video clip contained ten bicycles and four other vehicles. There were some recording problem that many frames were missing which was combined by some unusual bicyclist movements (as the bicyclists were aware of the experiment) to cause some tracking failures. However, all the lost bicyclists were re-detected later in the clip. There was only one misclassification (shown in the middle) but we did not have enough number of examples to provide any meaningful discussion on the classification performance. The resulting video clip can also be downloaded at the same website. Note that exactly the same grouping parameters were used for the result of this video clip and the one for Figure 5.

4.3. Interactive System to Generate High-quality Trajectories

There are many safety and traffic engineering applications that require a complete set of high-level trajectories. Most of them do not require on-line processing. Therefore, a vehicle tracking system that allows a minimal-level of user-interaction to validate and correct the detec-

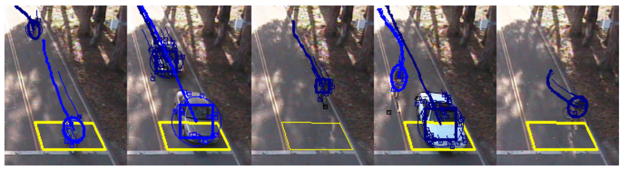


Figure 9. An application to bicycle detection and classification. The bicycles are shown in ellipses and other vehicles as squares. The yellow rectangular boxes are the virtual loops. One classification error (in the middle) and no detection failure from the video clip of ten bicycles and four other vehicles.

tion/tracking result can be quite useful [13]. We developed such an interactive vehicle detection system based on our algorithm that allows a user to locate an undetected vehicle, remove a false detection, reposition the center of the vehicle, modify trajectories, and even modify the corner-feature membership for a high-level user.

We used a a single-level grouping for the interactive system with a fixed vehicle size. When a user manually detects an object by specifying its position and the direction, a search is performed to find the member features – nearby corner features of the same motion.

The trajectory of an object is estimated from the velocity history and the position estimates (see Section 3.2). The trajectory estimation is illustrated in Figure 10. The first step is to correct the *position offset* given the accumulated velocity, shown in red. Note that the center position estimate in the first frame is not correct in many cases. For example, an object entering the scene will be only partially observable, and its center estimates will not be correct. Therefore, the system waits for a number of frames, say 10, to determine the position offset between the center estimates and the accumulated velocity ($t = \text{ref}_1$). When a user manually repositions the object early enough, the position offset is set at that frame.

The position offset is set and applied relatively to the heading of the object and the image resolution as shown as the dotted green line in Figure 10. Even if the position offset is correct, there still can be drifting while velocities are accumulated. The drift is not significant in most cases, but it can be large when the corner tracking is not reliable and/or the perspective change is too big (so that the pixel speed of the front end of the object is different from that of the rear end, for example).

The drift is set to 0 at or before the first reference frame. For any reference frame thereafter (via manually repositioned frames), the drift is set as the difference between center estimates and the position-offset-corrected point such that the final trajectory point becomes the center estimate. The drift between the reference frames are linearly interpolated. When the last reference frame is a manual modification, the drift is not changed thereafter. Otherwise, the drift is set such that the last trajectory position of the object be

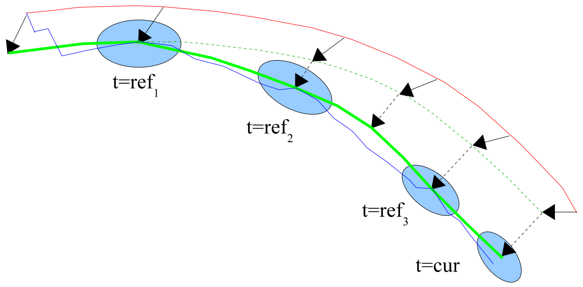


Figure 10. Estimation of the final trajectory. The red line denotes velocity accumulates, the blue line is for center estimates, and the dotted green line is the result of correcting position offset (solid arrow). The green line is the final trajectory after correcting drifts (dotted arrow).

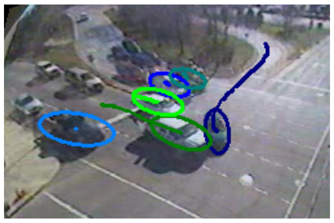


Figure 11. An example result of interactive detection. Automatic segmentation is extremely difficult in this case because the short video clip only contains the vehicles starting at the intersection with the same motion. Correct segmentation was possible only with several mouse clicks.

the same as the center estimate of the frame. The final trajectory after correcting the drift is shown as the solid green line.

We applied the interactive vehicle detection and tracking system to retrieve trajectories from seventy-five accident/incident video clips collected by the Kentucky Transportation Center, University of Kentucky [8]. An example result is shown in Figure 11. It is a short video clip where all vehicles stop at the intersection at the beginning of the video then move together as the traffic signal changes. Therefore, it is extremely difficult to segment the vehicles correctly. With a little aid (several mouse clicks) of a human operator, the system can extract the accurate positions and trajectories for all vehicles of interest. Example trajectories can be downloaded from the same website.

5. Summary and Future Work

We presented a new object detection and tracking approach that combines the background subtraction and the feature grouping algorithms. We introduced an augmentation to the background subtraction algorithm that uses a corner feature tracking as a cue, and a feature tracking and grouping algorithm that uses a background subtraction result as a cue. We introduced a dynamic multi-level feature grouping algorithm that can be applied to real time applications, handles various sizes of objects, and provides a set of robust trajectories. Promising results were presented with various transportation applications. The performance can be further improved by fusing multi-modal information such as by applying the vehicle classification result to constraining the size of the object and vice versa. Future work will include applying the algorithm to a larger number of data and performing comparative studies on various applications with various vision- and other sensor-based approaches.

6. Acknowledgment

This work was funded by multiple research projects at California Partners for Advanced Transit and Highways including CICAS Urban and Suburban Assisted Left-turn (USUAL) System (T.O. 6608), Bicycle Detection and Operation Concept at Signalized Intersections (T.O. 6203), System Engineering Management Plan (SEMP) for Loop Fault Detection (T.O. 6327), and Video-based Vehicle Trajectory Extraction for Intersection Crash Analysis (Toyota P.O. NATTC-0000025310).

References

[1] G. Antonini and J. P. Thiran. Counting pedestrians in video sequences using trajectory clustering. *IEEE Trans. Circuits and Systems for Video Technology*, 16(8):1008–1020, 2006.

[2] J. Batista, P. Peixoto, C. Fernandes, and M. Ribeiro. A dual-stage robust vehicle detection and tracking for real-time traffic monitoring. In *IEEE Intelligent Transportation Systems Conference*, pages 528–535, 2006.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real time computer vision system for measuring traffic parameters. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[4] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38, 1977.

[6] R. Ervin, C. MacAdam, J. Walker, S. Bogard, M. Hagan, A. Vayda, and E. Anderson. System for assessment of the vehicle motion environment (SAVME): volume i, ii, 2000.

[7] W. Forstner and E. Gulch. A fast operator for detection and precise location of distinct points, corners, and centers of circular features. In *Proc. Intercommision Conf. on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

[8] E. R. Green, K. R. Agent, and J. G. Pigman. Evaluation of auto incident recording system (AIRS). Research Report KRC-05-09, Kentucky Transportation Center, University of Kentucky, Sept. 2005.

[9] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Occlusion robust tracking utilizing spatio-temporal markov random field model. In *icpr*, volume 01, page 1140, 2000.

[10] N. K. Kanhere, S. J. Pundlik, and S. T. Birchfield. Vehicle segmentation and tracking from a low-angle off-axis camera. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[11] M. Kilger. A shadow handler in a video-based real-time traffic monitoring system. In *IEEE Workshop on Applications of Computer Vision*, 1992.

[12] Z. Kim. Realtime obstacle detection and tracking based on constrained delaunay triangulation. In *IEEE Intelligent Transportation Systems Conference*, pages 548–553, 2006.

[13] Z. Kim, G. Gomes, R. Hranac, and A. Skabardonis. A machine vision system for generating vehicle trajectories over extended freeway segments. In *12th World Congress on Intelligent Transportation Systems*, 2005.

[14] Z. Kim and J. Malik. Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. In *Proc. IEEE Intl. Conf. Computer Vision*, volume 1, pages 524–531, 2003.

[15] D. Koller, K. Daniilidis, and H.-H. Nagel. Model–based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[16] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. European Conf. on Computer Vision*, pages A:189–196, 1994.

[17] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *International Conference on Computer Vision*, pages 1154–1160, 1998.

[18] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[19] A. Worrall, G. Sullivan, and K. Baker. Pose refinement of active models using forces in 3d. In *Proc. 3rd European Conf. Computer Vision*, pages 341–352, 1994.