Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions

A. G. Amitha Perera Chukka Srinivas Anthony Hoogs Glen Brooksby Wensheng Hu
GE Global Research
One Research Circle, Niskayuna, NY 12309
perera@research.ge.com

Abstract

A fundamental requirement for effective automated analysis of object behavior and interactions in video is that each object must be consistently identified over time. This is difficult when the objects are often occluded for long periods: nearly all tracking algorithms will terminate a track with loss of identity on a long gap. The problem is further confounded by objects in close proximity, tracking failures due to shadows, etc. Recently, some work has been done to address these issues using higher level reasoning, by linking tracks from multiple objects over long gaps. However, these efforts have assumed a one-to-one correspondence between tracks on either side of the gap. This is often not true in real scenarios of interest, where the objects are closely spaced and dynamically occlude each other, causing trackers to merge objects into single tracks. In this paper, we show how to efficiently handle splitting and merging during track linking. Moreover, we show that we can maintain the identities of objects that merge together and subsequently split. This enables the identity of objects to be maintained throughout long sequences with difficult conditions. We demonstrate our approach on a highly challenging, oblique-view video sequence of dense traffic of a highway interchange. We successfully track the large majority of the hundreds of moving vehicles in the scene, many in close proximity, through long occlusions and shadows.

1. Introduction

As video becomes more and more pervasive in security and surveillance applications, it becomes more and more critical to have robust, automated analysis of object behavior and interactions to effectively process the large amount of data. This requires robust, automated tracking of objects, and, more importantly, it requires that objects are consistently identified through indefinite durations, despite long occlusions, sensing gaps, nearby traffic, pose changes, and other issues.

The vision community has mostly focused on tracking objects, and little work addresses explicitly the problem of maintaining identity. In general, the approach to maintaining identity has been to incorporate some concept of occlusion into the tracking algorithm. This enables tracking through short gaps, and thus maintains identity. However, these approaches fail when the gaps are even moderately long and the number of objects is high.

The main issue with tracking across long gaps is that the number of potential matches on the far side of the gap increases as the length of the gap increases. A common approach to address this problem is particle filtering, which maintains a probability density that captures the possible hypothesis. The expectation is that the likelihoods of the incorrect hypothesis will reduce over time, and they can thus be pruned away. Unfortunately, particle filtering is often only effective over short gaps; the search space becomes unmanageably large over long gaps. More recently, data association techniques have become more popular in tracking. In particular, multi-hypothesis tracking (MHT), developed initially for radar tracking, is another way of managing the hypothesis space. However, like particle filter algorithms, MHT algorithms suffer from a state space explosion when applied to real video tracking.

Another approach to solving the identity maintenance problem is to track objects until they become occluded, but not attempt to track across long gaps. Instead, the tracks are linked together at a higher processing level. This approach was used to identify vehicles in distributed cameras in highway monitoring [2, 5]. This track linking approach was also used to recover from tracking failures even within a single view in [8] and [3]. In both cases, short tracklets where linked together to form longer tracks, for automatically determining entry/exit points in the former, and for more robust tracking over long occlusions in the latter. One deficiency of the linking formulation of [3, 8] when applied to complex tracking scenarios is the assumption that a single tracklet tracks one object. In such scenarios, multiple objects are often merged into single detections, yielding a single track containing multiple objects. As objects merge and separate, the tracker must be able to merge and split the tracks to compensate. Some work has been done in incorporating merge and split hypothesis into the data association based trackers [4], but they have been developed for scenarios where the objects are continuously visible. Moreover, the identities of the objects are lost (confused) when objects merge and subsequently split. To our knowledge, the splitting and merging of tracks has not been addressed with the track linking approach.

This paper describes a technique that handles merges and splits in the track linking approach to maintaining identity. We follow the general scheme of [3], but we use different algorithms in the moving object detection and tracking layers, and overhaul the track linking layer to cope with merges and splits. An additional innovation in our work is that we show how to maintain the identity of objects through merges and subsequent splits.

Overall, our approach provides a fully automated tracker that correctly identifies objects in the sequence through occlusions and merges with other objects. Through the track linking framework, we can handle the difficult cases of track merges and splits that occur just before and after long occlusions. We demonstrate this on difficult aerial surveillance video of a highway interchange. We quantify our results against ground truth and show improvement over (1) tracking without linking and (2) tracking with one-to-one linking.

Following [3], the overall system consists of three parts, which are described in the following sections.

1. Moving object detection. We use the Stauffer-Grimson algorithm [9].

2. Automatically initialized tracking. We use a nearest-neighbor data association tracker to track objects based on the output of step 1.

3. Track linking with merges and splits. We link the fragmented tracks output by 2 to generate long object tracks through inaccurate moving object segmentations. This is the main contribution of this paper.

2. Moving object detection

We use the Stauffer-Grimson background modeling algorithm [9] to detect moving objects in the scene. This algorithm models the intensity of each pixel as a mixture of Gaussians, and flags a pixel as belonging to a moving object when it does not match the modes that correspond to the background intensities. Since our sensor is moving, we stabilized the video using homographies estimated from KLT [7] features before applying background modeling. After stabilization, the scene is essentially stationary except for parallax-induced motion.

This approach produces far fewer false alarms and artifacts compared to the frame differencing approach of [3], but does require a much longer sensor dwell time: it takes about 30 frames to initialize the background model.

We use simple morphology to separate the foreground pixel mask into separate moving object detections. Since vehicles are compact, mostly convex objects, we fill each detection with the convex hull to further reduce pixel-level noise.

Ideally, each detection contains a single object, but many contain multiple objects if they are visually close or overlapping.

3. Tracking

We use a simple nearest-neighbor data association tracker to generate the basic tracks. Since our focus is on the subsequent track linking, we did not experiment with more complex trackers. Note, however, that (1) our overall results are quite good even with this simple tracker (section 5), and (2) it is more difficult to derive good termination conditions with more complex trackers. The latter point is important for the track linking approach, since it expects conservative tracks.

Tracking Each tracker maintains a Kalman filter with a state consisting of position and velocity. At each step, each tracker produces a validation gate and associates the detection with the gate that is nearest to the predicted position. Each tracker then updates its state using the associated detection.

Termination Trackers are terminated for one of two reasons. First, if a tracker is not associated with a detection for n consecutive frames ($n = 3$ in our experiments), the tracker is terminated. Second, if two or more trackers associate with the same detection, all of them are terminated, and the detection is marked as unassociated. Terminated tracks are recorded in a database for later linking.

Initialization Each unassociated detection launches a new tracker. These trackers are marked as "initializing" for m frames ($m = 3$ in our experiments). Being marked as initializing has two consequences for a tracker. First, it has stronger conditions for termination (for example n above is 1 for initializing tracks). Second, if an initializing track is terminated, it is *not* recorded in the database; it is as if they never occurred. If a tracker survives for m frames, it is marked as "formed". The initializing phase greatly reduces the number of false alarm tracks.

4. Track linking

4.1. One-to-one correspondence

Following [3, 8], we formulate the track linking problem by computing pairwise linking costs and using the Hungarian algorithm to solve the resulting assignment problem. This formulation makes the assumption that joint probability of associating tracks with an object can be decomposed as the product of pairwise probabilities. That is,

$$\mathrm{P}(\{T_1, \ldots, T_n\}) = \prod_{i=1}^{n-1} \mathrm{P}_{\text{1-1}}(T_i \vdash T_j), \tag{1}$$

where $\mathrm{P}(\{a, b, c\})$ is the probability that tracks a, b, and c are associated with the same object.

We define the pairwise probability of associating two tracks as the product of three components (temporal, kinematic, and appearance):

$$\mathrm{P}_{\text{1-1}}(T_i \vdash T_j) = \mathrm{P}_{\mathrm{t}}(T_i \vdash T_j)\, \mathrm{P}_{\mathrm{k}}(T_i \vdash T_j)\, \mathrm{P}_{\mathrm{a}}(T_i \vdash T_j). \tag{2}$$

Temporal P_{t} captures the temporal nature of the association:

$$\mathrm{P}_{\mathrm{t}}(T_i \vdash T_j) = \begin{cases} 1 & \text{if } e_i < s_j, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $e_i = \mathrm{end}(T_i)$ and $s_j = \mathrm{start}(T_j)$.

Kinematic P_{k} captures the kinematics of the tracks. Each track T_i has a set of tuples (x_k^i, y_k^i, t_k^i) indicating the location of the object at time t_k^i. We use multi-variate regression to estimate with a linear model to this tuples, using t as the independent variable. This corresponds to a constant velocity estimate of the motion. The regressions provides both the model parameters and parameter covariance matrix. These can be used to obtain an estimate of the object position $\hat{\boldsymbol{x}}^i(t')$ and error covariance $\hat{\boldsymbol{\Sigma}}^i(t')$ at any time t'. To predict forward in time (past the end of the track), we use an estimate based on the last n frames. To predict backward, we use an estimate based on the first n frames. We denote the forward estimates by $\hat{\boldsymbol{x}}_{\mathrm{f}}^i(\cdot)$ and $\hat{\boldsymbol{\Sigma}}_{\mathrm{f}}^i(\cdot)$, and the backward estimates by $\hat{\boldsymbol{x}}_{\mathrm{b}}^i(\cdot)$ and $\hat{\boldsymbol{\Sigma}}_{\mathrm{b}}^i(\cdot)$. In our experiments, $n = 15$, corresponding to 2.5 seconds. We define

$$\begin{aligned} \mathrm{P}_{\mathrm{k}}(T_i \vdash T_j) =& \mathrm{P}_{\mathrm{gn}}(\hat{\boldsymbol{x}}_{\mathrm{b}}^j(s_j); \hat{\boldsymbol{x}}_{\mathrm{f}}^i(s_j), \hat{\boldsymbol{\Sigma}}_{\mathrm{f}}^i(s_j), \sigma_{\mathrm{g}}^2) \times \\ & \mathrm{P}_{\mathrm{gn}}(\hat{\boldsymbol{x}}_{\mathrm{b}}^i(e_i); \hat{\boldsymbol{x}}_{\mathrm{b}}^j(e_i), \hat{\boldsymbol{\Sigma}}_{\mathrm{b}}^j(e_i), \sigma_{\mathrm{g}}^2), \end{aligned} \tag{4}$$

where σ_{g} is a parameter and P_{gn} is a gated normal distributed defined by

$$\mathrm{P}_{\mathrm{gn}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma_{\mathrm{g}}^2) = \frac{f(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma_{\mathrm{g}}^2)}{\int_{\boldsymbol{x}'} f(\boldsymbol{x}'; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma_{\mathrm{g}}^2)}, \tag{5}$$

with

$$f(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma_{\mathrm{g}}^2) = g\big((\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}), \sigma_{\mathrm{g}}^2\big) \tag{6}$$

and

$$g(x^2, \sigma_{\mathrm{g}}^2) = \begin{cases} \exp(-\frac{1}{2}x^2), & x^2 < \sigma_{\mathrm{g}}^2, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Appearance We define the appearance probability using template matching, which is a stronger appearance model than the color histogram matching in [3] or the mean color matching in [2]. To associate T_i and T_j, we estimate an appearance template A_i for T_i using the last few frames of T_i, and a template A_j using the first few frames of T_j. Further, we estimate the orientations of the objects as $\hat{\theta}_i = \frac{\partial}{\partial t} \hat{\boldsymbol{x}}_{\mathrm{f}}^i(e_i)$ and $\hat{\theta}_j = \frac{\partial}{\partial t} \hat{\boldsymbol{x}}_{\mathrm{b}}^j(s_j)$. Then, we define

$$\mathrm{P}_{\mathrm{a}}(T_i \vdash T_j) = \mathrm{corr}\big(A_i, R(A_j, \hat{\theta}_i - \hat{\theta}_j)\big), \tag{8}$$

where $R(A, \theta)$ rotates the template A by θ.

4.2. Merges and splits

Suppose two objects A and B merge for a while, so that

$$\begin{aligned} A =& \{T_{a,1}, \ldots, T_{a,m}, T_{c,1}, \ldots, T_{c,o}, T_{d,1}, \ldots, T_{d,p}\} \\ B =& \{T_{b,1}, \ldots, T_{b,n}, T_{c,1}, \ldots, T_{c,o}, T_{e,1}, \ldots, T_{e,q}\}. \end{aligned} \tag{9}$$

Using the pairwise assumption,

$$\begin{aligned} \mathrm{P}(A, B) =& \mathrm{P}(\{T_{a,1}, \ldots, T_{a,m}\}) \times \mathrm{P}(\{T_{b,1}, \ldots, T_{b,m}\}) \times \\ & \mathrm{P}_{\mathrm{m}}(T_{a,m}, T_{b,n} \vdash T_{c,1}) \times \mathrm{P}(\{T_{c,1}, \ldots, T_{c,o}\}) \times \\ & \mathrm{P}_{\mathrm{s}}(T_{c,o} \vdash T_{d,1}, T_{e,1}) \times \mathrm{P}(\{T_{d,1}, \ldots, T_{d,p}\}) \times \\ & \mathrm{P}(\{T_{e,1}, \ldots, T_{e,q}\}) \end{aligned} \tag{10}$$

with P_{m} and P_{s} defining the probability of a merge and split, respectively. As with the one-to-one case, we define each of these as the product of three components. For merges, we define

$$\begin{aligned} \mathrm{P}_{\mathrm{m}}(T_i, T_j \vdash T_k) =& \mathrm{P}_{\mathrm{t,m}}(T_i, T_j \vdash T_k) \times \mathrm{P}_{\mathrm{k,m}}(T_i, T_j \vdash T_k) \times \\ & \mathrm{P}_{\mathrm{a,m}}(T_i, T_j \vdash T_k) \end{aligned} \tag{11}$$

with

$$\mathrm{P}_{\mathrm{t,m}}(T_i, T_j \vdash T_k) = \begin{cases} 1, & \text{if } e_i < s_k \text{ and } e_j < s_k, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

$$\begin{aligned} \mathrm{P}_{\mathrm{k,m}}(T_i, T_j \vdash T_k) =& \mathrm{P}_{\mathrm{gn}}(\hat{\boldsymbol{x}}_{\mathrm{b}}^k(s_k); \hat{\boldsymbol{x}}_{\mathrm{f}}^i(s_k), \hat{\boldsymbol{\Sigma}}_{\mathrm{f}}^i(s_k), \sigma_{\mathrm{g}}^2) \times \\ & \mathrm{P}_{\mathrm{gn}}(\hat{\boldsymbol{x}}_{\mathrm{b}}^k(s_k); \hat{\boldsymbol{x}}_{\mathrm{f}}^j(s_k), \hat{\boldsymbol{\Sigma}}_{\mathrm{f}}^j(s_k), \sigma_{\mathrm{g}}^2), \end{aligned} \tag{13}$$

$$\mathrm{P}_{\mathrm{a,m}}(T_i, T_j \vdash T_k) = \mathrm{P}_{\mathrm{a}}(T_i \vdash T_k)\, \mathrm{P}_{\mathrm{a}}(T_i \vdash T_k). \tag{14}$$

Observing that a split is essentially a merge reversed in time, we compute P_{s} exactly the same way as P_{m} except that the time axis is temporarily reversed.

4.3. Finding the assignment

For the one-to-one correspondence case with n tracks, finding the optimal assignment is straight-forward: form an $n \times n$ cost matrix $C = \{C_{ij}\}$ with

$$C_{ij} = -\log \mathrm{P}_{\text{1-1}}(T_i \vdash T_j) \tag{15}$$

and apply the Hungarian algorithm. Tracing through the pairwise assignments then yields a linked track.

If we hypothesize a merge of, say, T_a and T_b with T_c, we can form a new matrix C' exactly the same as above, delete rows a and b, and add a new row g such that

$$C'_{gi} = \infty \quad \forall i \neq c, \text{ and} \tag{16}$$

$$C'_{gc} = -\log \mathrm{P}_{\mathrm{m}}(T_a, T_b \vdash T_c). \tag{17}$$

If the cost of the solution to C' is less than the cost of the solution to C, then T_a and T_b did merge, and the solution to C' is the final solution; if not, the tracks did not merge and the solution to C is the final solution.

Unfortunately, with this approach, if we hypothesize m merges, then we have to solve 2^m assignment problems to find the final solution. This will be prohibitively expensive in practice, so we describe a suboptimal algorithm that works well in practice.

Suppose we have m merge hypotheses

$$M = \{(T_{a_i}, T_{b_i} \vdash T_{c_i})\}, \quad i = 1, \ldots, m. \tag{18}$$

We write $M = \{(a_i, b_i, c_i)\}$ for notational convenience. First, we form the standard one-to-one correspondence cost matrix C. Next, for each distinct merge target c_i we add a column to C, labeled c'_i. These columns are initially filled with ∞. Denote this augmented matrix C^0. Finally, for each hypothesized merge $(a, b, c) \in M$, set $C^0_{bc'} = C^0_{ac} + \log \mathrm{P}_{\mathrm{m}}(T_a, T_b \vdash T_c)$. Observe that if the solution to C^0 assigns row a to column c, row b to column c', and each other row to a column $\leq n$, then the cost of this solution is exactly the cost of C' above, since

$$C^0_{ac} + C^0_{bc'} = -\log \mathrm{P}_{\mathrm{m}}(T_a, T_b \vdash T_c) = C'_{gc}. \tag{19}$$

This is true for each merge hypothesis that was added to the matrix. However, not every assignment corresponds to a valid solution. For example, if row b is assigned to column c', but row a is not assigned to column c, then that solution cannot be interpreted as a merge case. We then have to modify C^0 somehow to find a valid solution. Our algorithm is as follows:

1. $t \leftarrow 0$
2. Let A be the optimal assignment to C^t, such that $A(i) = j$ means that row i is assigned to column j.
3. If $A(b_i) = c'_i \Rightarrow A(a_i) = c_i \quad \forall (a_i, b_i, c_i) \in M$, stop. A is a valid solution.
4. Otherwise, let $M' = \{(a_i, b_i, c_i) \in M : A(b_i) = c'_i \wedge A(a_i) \neq c_i\}$.
5. Find k such that $C_{ik} = \min_{(a,b,c) \in M'} C_{bc'}$.
6. Let C^{t+1} be the same as C^t with column k set to ∞. This removes a merge to T_k from the hypothesis set.
7. $t \leftarrow t + 1$. Goto step 2.

To handle splits, the procedure is similar; we just augment the one-to-one correspondence matrix with additional rows.

4.4. Maintain identity through merges

Once we hypothesize and confirm a merge $(T_a, T_b \vdash T_c)$, we know that T_c is a combination of tracks T_a and T_b. If T_c subsequently splits into T_d and T_e, we do not know if T_a goes with T_d or with T_e. There are two ways to handle this. One is to evaluate the two hypothesis $\{(T_a, T_d), (T_b, T_e)\}$ and $\{(T_a, T_e), (T_b, T_d)\}$ and choose the better one. The other is to use T_a and T_b to explicitly split T_c into two tracks $T_{a'}$ and $T_{b'}$, and then do a normal one-to-one correspondence with all six tracks T_a, T_b, $T_{a'}$, $T_{b'}$, T_d, and T_e. We chose to implement the latter.

Our final track linking algorithm is as follows.

1. Hypothesize a set of merges and splits.
2. Generate and solve the augmented matrix.
3. If the solution does not have any merges, stop.
4. Otherwise, split each merge target, and go to 1.

5. Results

We validated our approach on a difficult aerial surveillance scenario. Figure 2 shows examples from the sequence. The video was taken from a slow moving platform with a large format camera acquiring images at 6Hz. The sequence was stabilized after acquisition by using KLT [7] features to compute frame to frame homographies. The sequence captures 162 vehicles moving on a multi-lane highway interchange. On average, there are 42 vehicles in each frame. Many of the vehicles traveling on the main highways dynamically occlude each other. Some of the vehicles always appear merged throughout their presence in the sequence. (For example, a car moving next to a truck at the same speed.) Figure 3 shows the tracks overlaid on the image frame. Each track is colored with a different color. While it is hard to tell the difference because of the sheer number of objects, the image on the right (the linked result) appears somewhat smoother and uniform. This is because of the long, linked tracks generated by our track linking algorithm.

To evaluate our algorithms, we manually generated ground truth tracks for each vehicle. To associate generated tracks T_i with ground truth tracks G_j, we used the following track distance measure [6]:

$$D(T_i, G_j) = \frac{1}{|O(T_i, G_j)|} \sum_{t \in O(T_i, G_j)} \|\boldsymbol{x}_t^i - \boldsymbol{x}_t^j\|, \quad (20)$$

where $O(T_i, G_j)$ denotes the frames where T_i and G_j overlap, $|\cdot|$ denotes cardinality, and $\|\cdot\|$ denotes the L2-norm. An association A is a set of pairs (i, j) such that $(i, j) \in A$ iff T_i is associated with G_j. Abusing notation, we define $A(G_j) = \{T_i | (i, j) \in A\}$. The performance metrics below are computed w.r.t. the optimal association

$$A^* = \arg\min_A \sum_{(i,j) \in A} D(T_i, G_j) \quad (21)$$

with the minimization subject to

$$O(T_i, T_k) = \emptyset \quad \forall (i, j), (k, j) \in A, \quad (22)$$

which ensures that two temporally overlapping generated tracks are not assigned to the same ground truth track.

With this association in hand, we use a few metrics from the tracking performance evaluation literature [1, 6, 10] to measure our performance.

The first metric we use is the *object detection rate*, which is

$$\text{ODR} = \frac{\text{\# correct detections}}{\text{\# frames in track}}. \quad (23)$$

Note that this is independent of the track association above. This metric measures how well we can determine that there is an object at a given location at a given time. A related metric is the *track completeness factor*,

$$\text{TCF} = \frac{\sum_i \sum_{T_j \in A(G_i)} |O(T_j, G_i)|}{\sum_i |G_i|}. \quad (24)$$

This measures how well we detect a given object after the association. (That is, with some concept of object identity.) Note that, in both cases, we do not compute the metrics per track and average, since that would bias the overall metric toward short tracks. Finally, the two most important metrics for measuring how well we maintain identity are the *track fragmentation*

$$\text{TF} = \frac{\sum_i |A(G_i)|}{|\{G_i | A(G_i) \neq \emptyset\}|}, \quad (25)$$

and its normalized version,

$$\text{NTF} = \frac{\sum_i |G_i| \cdot |A(G_i)|}{\sum_{i | A(G_i) \neq \emptyset} |G_i|}. \quad (26)$$

	ODR	TCF	TF	NTF
No Interp				
Unlinked	0.702	0.696	3.066	3.543
Kine	0.702	0.611	1.230	1.269
Kine & App	0.702	0.618	1.255	1.299
KAMS unlinked	0.706	0.697	3.098	3.590
KAMS Kine & App	0.706	0.624	1.174	1.221
Interp				
Unlinked	0.702	0.696	3.066	3.543
Kine	0.810	0.692	1.209	1.246
Kine & App	0.805	0.700	1.242	1.285
KAMS unlinked	0.706	0.697	3.098	3.590
KAMS Kine & App	0.820	0.726	1.168	1.217

Table 1. Overall measures of performance. See the text for a description of the metrics. Track linking improves the fragmentation score (TF) significantly (implying better identity maintenance).

The normalization increases the weight for longer tracks, to account for the fact that it is more difficult (and important) to maintain the identity of long tracks than short ones. A fragmentation score of n means that we have identified the object with n labels (tracks). A fragmentation of 1 is the ideal.

Table 1 shows results for various flavors of the algorithm. In the table, the "no interp" results simply link the tracks into a single long track with gaps in the middle, while the "interp" results interpolates across the gaps. "Unlinked" scores the basic tracking result. "Kine" and "Kine & App" shows the results when only kinematics and both kinematics and appearance, respectively, are used to link the tracks. "KAMS" are the tracks resulting from the merge and split (MS) processing using the hypothesis generated using kinematic and appearance (KA) linking. These tracks can be left as is ("KAMS unlinked") or linked again using kinematics and appearance ("KAMS Kine & App").

Without interpolation or merge and split processing, the ODR is constant, because the tracks, and therefore the detected positions, do not change. Both interpolation and merge and split processing add new detections (the former by filling the gaps between tracks, the latter by creating new tracks), and thus increase the ODR.

The key advantage of the track linking is the drop in the track fragmentation. The fragmentation drops from 3.1 per track without linking to only 1.2 per track when linking with merge and split processing. This shows that we are able to link the fragmented tracks back together to recover the object identity.

The track completion factor for the unlinked case is the best possible score given the generated tracks, because it essentially corresponds to linking with ground truth. When we link, we are trading off the completion factor against the fragmentation. The results show that the linking more than halves the fragmentation score without a great loss in the

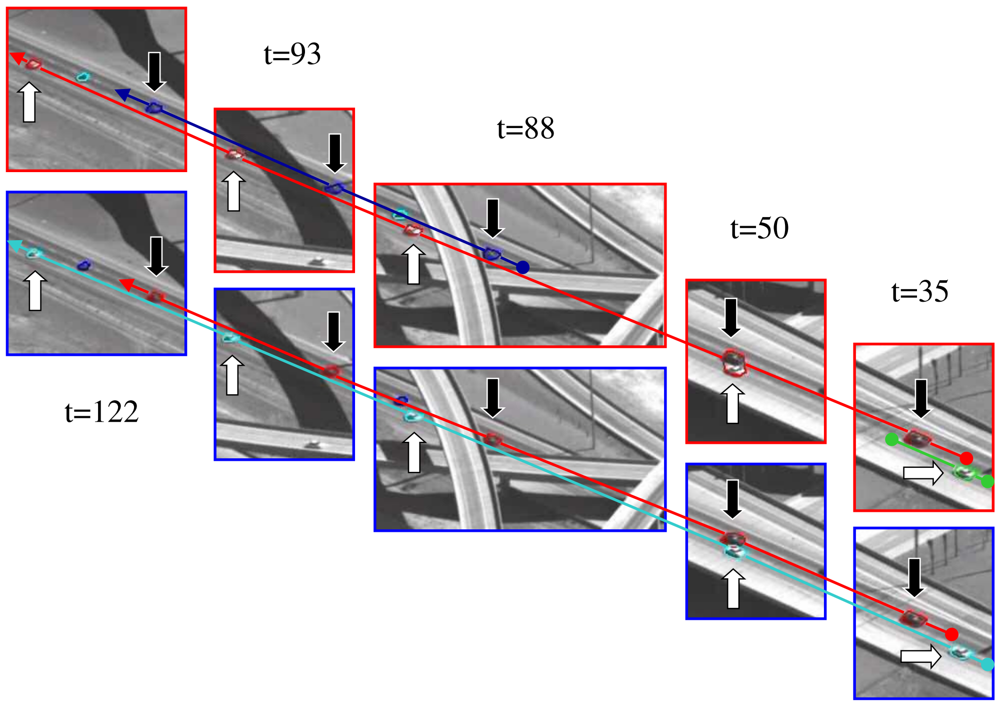


Figure 1. These illustrate the merge and split processing. The images on top (red border) show linking without merge processing, and the images underneath (blue border) show linking after merge processing. The black and white arrows point to the same black and white cars over time. The solid lines show the linking results, with the dots indicating start and end points. After merge processing, the merged track (illustrated in t=50) is split into two objects, allowing the previously terminated white car track (top, green) to continue all the way through (bottom, cyan). Thus, the identity is maintained through the merge condition and through the long occlusion under the bridges (shown at t=88). See the text for more detail.

completeness factor. In fact, when we interpolate across the gaps, there is no loss in the completion factor for the gain in fragmentation. The conclusion is that we manage to effectively maintain the identity of the objects. The merge and split processing manages to further reduce the fragmentation because some of the merged tracks are successfully split into multiple objects, enabling linking through the merge.

The latter point is illustrated in Figure 1. Initially, the black and white cars are tracked separately (t=35). At some point, those tracks stop, and a merged track is initiated as the white car passes the black car (t=50). Soon after that, the cars pass under the bridges, and new, separate tracks are initiated when they reappear (t=88,93). Without the merge processing (top, red border), the one-to-one condition forces one of the initial tracks (t=35) to be dropped at the merge (t=50); in this case, the white car track is dropped. The single merged track is then linked to one of the subsequent black or white car tracks (t=88). Here, it happened to link to the white car, causing an identity switch. However, with the merge processing (bottom, blue border), the two initial tracks (t=35) are hypothesized to merge (t=50). The hypothesis is validated, and the merged track is split into two tracks. This allows the normal one-to-one linking to correctly maintain the identity of both the black and white cars.

6. Conclusions

Handling splits and merges during tracking is a significant, ongoing challenge. By casting the problem within the track linking framework, we demonstrate how it can be managed efficiently without exploding the hypothesis space while maintaining a reasonable approximation to the global, multi-object solution. Our current focus is to explore in more detail the interactions of the kinematic and appearance costs, especially in the merge and split cases.

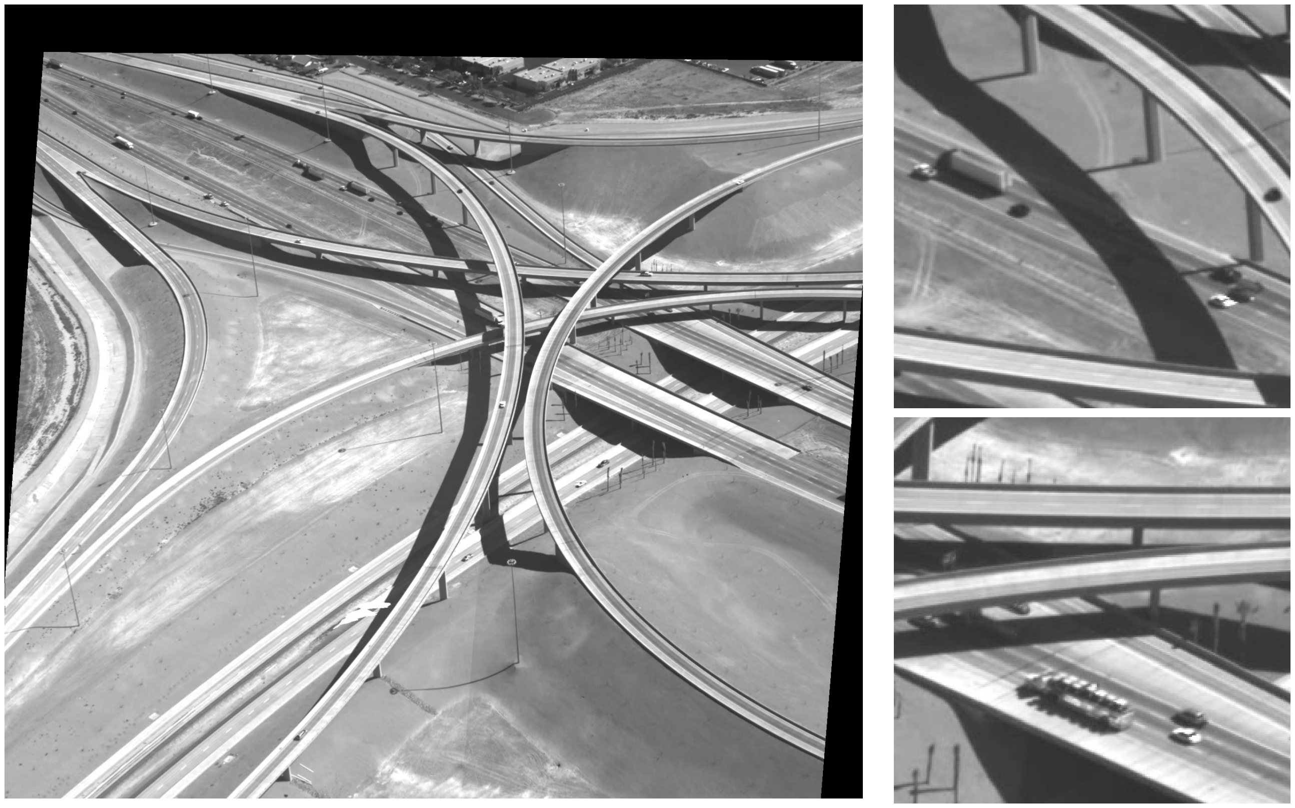


Figure 2. Examples from the sequence.

7. Acknowledgments

We wish to Gianfranco Doretto for assisting us in implementing some of the ground truth association and performance metric computation code.

This report was prepared by GE GRC as an account of work sponsored by Lockheed Martin Corporation. Information contained in this report constitutes technical information which is the property of Lockheed Martin Corporation. Neither GE nor Lockheed Martin Corporation, nor any person acting on behalf of either; a. Makes any warranty or representation, expressed or implied, with respect to the use of any information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or b. Assume any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method or process disclosed in this report.

References

[1] J. Black, T. Ellis, and P. Rosin. A novel method for video tracking performance evaluation. In *PETS*, pages 125–132, 2003.

[2] T. Huang and S. Russell. Object identification: A Bayesian analysis with application to traffic surveillance. *Artifical Intelligence*, 103, 1998.

[3] R. Kaucic, A. G. A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *Proc. CVPR*, pages 990–997, 2005.

[4] Z. Khan, B. Tucker, and F. Dellaert. Multitarget tracking with split and merged measurements. In *Proc. CVPR*. IEEE, 2005.

[5] H. Pasula, S. Russel, M. Ostland, and Y. Ritov. Tracking many objects with many sensors. In *Int. Joint. Conf. on Art. Intel.*, pages 1160–1171, Stockholm, Sweden, 1999.

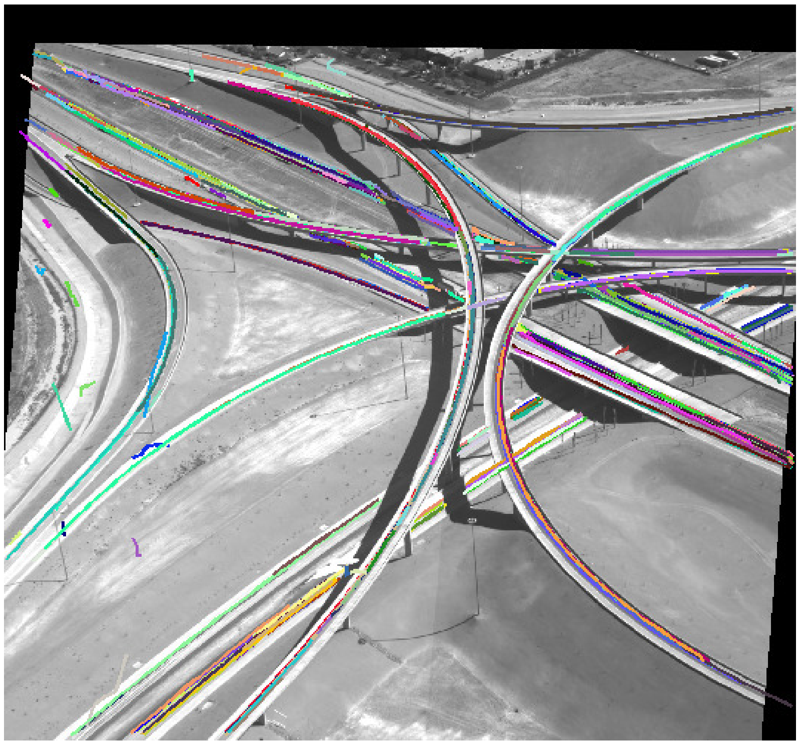
[6] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. In *Proc. IEEE Int. Workshop on PETS*, Kauai, Hawaii, USA, 9 Dec. 2001.

[7] J. Shi and C. Tomasi. Good features to track. In *Proc. CVPR*, pages 593–600, 1994.

[8] C. Stauffer. Estimating tracking sources and sinks. In *Proceedings of the IEEE Workshop on Event Mining in Video*, 2003.

[9] C. Stauffer and E. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. CVPR*, pages 246–252, 1999.

[10] A. Theil, R. A. W. Kemp, K. Romeo, L. J. H. M. Kester, and É. Bossé. Classification of moving objects in surveillance algorithms. In *Proc. IEEE Int. Workshop on PETS*, Grenoble, France, 31 Mar. 2000.



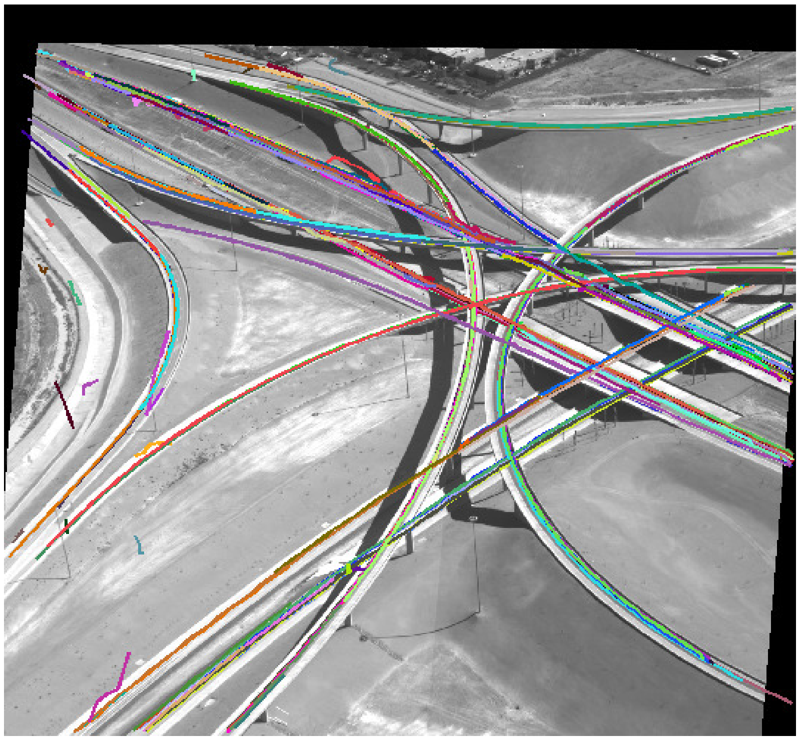


Figure 3. These two images illustrate the tracks before and after linking. It is somewhat difficult to see the difference simply because there is so much traffic, but the linked version (on the right) has a smoother color distribution resulting from the long, linked tracks.