Inferring Tracklets for Multi-Object Tracking

Jan Prokaj[1], Mark Duchaineau[2], and Gérard Medioni[1]

[1]University of Southern California , Los Angeles, CA 90089 , {prokaj|medioni}@usc.edu
[2]Lawrence Livermore National Laboratory , Livermore, CA 94550, duchaineau@llnl.gov

Abstract

Recent work on multi-object tracking has shown the promise of tracklet-based methods. In this work we present a method which infers tracklets then groups them into tracks. It overcomes some of the disadvantages of existing methods, such as the use of heuristics or non-realistic constraints. The main idea is to formulate the data association problem as inference in a set of Bayesian networks. This avoids exhaustive evaluation of data association hypotheses, provides a confidence estimate of the solution, and handles split-merge observations. Consistency of motion and appearance is the driving force behind finding the MAP data association estimate.

The computed tracklets are then used in a complete multi-object tracking algorithm, which is evaluated on a vehicle tracking task in an aerial surveillance context. Very good performance is achieved on challenging video sequences. Track fragmentation is nearly non-existent, and false alarm rates are low.

1. Introduction

Object tracking is a well studied problem that has received considerable attention in the computer vision community, and does not need much of an introduction. Solving this problem will immediately benefit many applications, such as surveillance and human computer interaction, and provide necessary information with high confidence for higher-level reasoning tasks, such as activity or event recognition, and behavior analysis. Many approaches have been proposed [17], each having advantages in a particular context. Here we focus on multi-object tracking, where the goal is to determine a spatio-temporal description of the moving objects in the scene.

Tracking of multiple objects naturally has to deal with the same issues encountered in classic tracking, such as occlusion, and changes in illumination and appearance. It also brings additional challenges, namely associating the evidence in the video with an unknown number of objects, and object-object interaction, which results in a many-many mapping between observations (detections or measurements) and objects in every frame. In other words, one observation may be associated with multiple objects, or multiple observations may be associated with one object due to occlusion or noise. This presents a large data association problem, which is expensive to solve optimally: a correct data association requires looking ahead in time, but this in turn causes an exponential growth of the search space.

Currently the most successful algorithms for multi-object tracking solve the data association problem hierarchically [13, 2, 6, 8]. They first determine short tracks, or tracklets, and then link these tracklets into longer tracks in one or more steps. Tracklets are usually determined by using a nearest neighbor association [13], some affinity measure [6], or particle filtering [8]. However, each of these has some disadvantages. Nearest neighbor association is a heuristic that is likely to make incorrect associations when objects are close together. Affinity measures, while conservative, assume a non-overlap constraint. Particle filtering works better, but with a lot of objects or noise in the scene, and longer time intervals, the number of hypotheses significantly increases, and the computational cost becomes prohibitive. An alternative solution is proposed in [15] where the scene is first divided into grid cells and the Hungarian algorithm is then used to estimate the association of detections from one frame to the next. One disadvantage here is that weights in the bipartite graph matching problem are based on local context that is assumed to stay constant, which may not work as well in urban areas where vehicles make turns.

The primary contribution of this work is an algorithm for inferring tracklets, which does not require an exhaustive evaluation of data association hypotheses, is a MAP estimate, rather than a heuristic, does not assume one-one mapping between observations and objects, and provides a confidence measure on each tracklet. The algorithm accomplishes this by formulating the problem as inference in a set of Bayesian networks, and uses consistency of motion and appearance as the driving force. The computed tracklets are then used in a complete multi-object tracking algorithm, which is evaluated on a vehicle tracking task in an aerial surveillance context.

2. Related Work

The classic algorithms for multi-object tracking are Joint Probabilistic Data Association filter (JDPAF) [4], and Multiple Hypothesis Tracking (MHT) [14, 3]. Both of these methods operate on a set of data association hypotheses. JPDAF uses all hypotheses, weighted according to the posterior, in determining the next state, while MHT keeps the candidate hypotheses in memory and does not make a decision until any ambiguities are resolved. In addition to requiring exponential space and time, a significant weakness of these algorithms is that they assume one-one mapping between observations and objects.

More useful approaches that handle occlusions in multi-object tracking are [19] and [7]. In [19], a globally optimal data association was achieved by encoding the data association problem in a network flow graph, and solving with a minimum cut. While this strategy is compelling, the method assumes that two tracks cannot have overlapping observations, which happens with merged observations. A linear programming formulation, which allows for occlusions, but assumes spatial layout constraints, was presented in [7].

There have been several attempts to handle multiple split-merge events in observations. A particle filtering approach was presented in [9]. There, a more efficient solution is obtained by assuming Gaussian motion, and computing the target state analytically for each sampled data association. A particle filtering approach using Adaboost for more robust detections is presented in [12]. Another sampling algorithm, using data-driven MCMC, was introduced in [18]. Spatio-temporal smoothness in motion and appearance was key to recovering the tracks of an unknown number of targets.

Perera et al. described an algorithm for multi-object tracking that handles long occlusions, and split-merge conditions in [13]. They begin with detecting a set of tracklets, and subsequently link these tracklets into long tracks. Linking is accomplished with the Hungarian algorithm and using an approximation that does not require an exhaustive search of associations. This algorithm requires that the initial set of tracklets be reliable. More tracklet-based approaches to multi-object tracking followed in [6, 8]. An affinity measure to determine tracklets was used in [6], and a particle filter in [8].

It is clear that tracklet-based approaches have merit, but as mentioned earlier, the current algorithms to determine tracklets have some disadvantages. In this work we present an approach to infer the tracklets that tries to overcome these disadvantages.

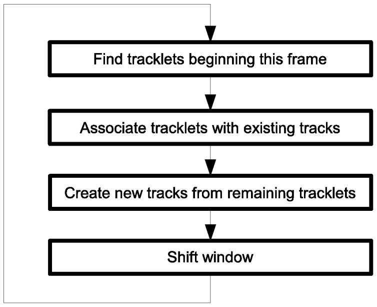

Find tracklets beginning this frame
Associate tracklets with existing tracks
Create new tracks from remaining tracklets
Shift window

Figure 1: The role of tracklets in multi-object tracking. The first step is the main focus of this paper.

3. Approach

The goal of our algorithm is to infer tracklets, each representing one object, over a (sliding) window of frames. This window is usually 4-8 seconds in length. The input to our algorithm is a set of object detections (blobs) in each frame. These can be as simple as connected components taken directly from background subtraction, or they can be the output of a more complex object detector. Each object detection also has an associated appearance representation, such as the raw image patch, or a histogram.

We would like to emphasize that our goal is to find valid tracklets within a window that shifts with each frame. Aggregation of these tracklets into tracks that span several windows is done by the (higher-level) tracking algorithm. Also if the detections of an object become split (or merged) for a period longer than the window size, this algorithm will find several (or one) tracklets in the window. This must be handled at the higher-level as well. An example of a multi-object tracker that takes care of these issues is in Section 4. A flow-chart that clarifies multi-object tracking using tracklets is in Figure 1.

We do not assume an a priori number of objects in the scene, and the number can vary over time. Determining the tracklets optimally with this relaxation is difficult, because the number of possible associations is unknown. The traditional way to handle this is to sample the space of different object counts, or try a different number of objects in an iterative fashion, and use an energy function to determine the answer. Unfortunately, unless simplifying assumptions are made, this energy function is nonconvex, and there may be multiple local minima that one get trapped in.

To avoid this problem, we assume that each detection in the first frame of the window is a potential object. Therefore, we find an optimal tracklet, or a set of tracklets, starting at each detection in the first window frame. This is not a problem, because for detections that are false alarms, the model of a valid tracklet (consistency of motion and appearance) is not satisfied, and the tracklet is discarded. Tracklets that start in the second or later frame of the window are found when the sliding window shifts to that frame.

3.1. Problem formulation

If the initial detection of an object is given to us, we know there must be another detected instance of that object located "nearby" in subsequent frames. We are assuming there are no missed detections (due to occlusion or else) for now, but split detections, or split-merge detections, do not pose a problem, and this statement still holds. Therefore, the optimal tracklet, or a set of tracklets, that we want to find must be composed of a series of "nearby" detections. This can be expressed in a detection tree. For a window size of T frames, this tree would have T levels. A node in level t has links to those nodes in level $t+1$, which are "nearby." The root of the tree, $t = 0$, represents the initial detection. The definition of nearby can vary with dataset, but, for example, in aerial surveillance of urban areas, this value can be set according to the maximum expected velocity of vehicles, and video resolution.

The number of possible tracklets and tracklet combinations arising from this detection tree is huge, and we certainly do not want to evaluate every hypothesis. Instead, we realize that every such hypothesis is making a decision about including or not including each detection. In other words, this is just a binary labeling, or segmentation problem. The valid detections need to be separated from the invalid detections. The invalid detections are detections due to noise, or due to objects other than the one that generated the initial detection. One consequence of this view is that given the valid detections, it is not always known which tracklets generated them. For example, when there are multiple valid detections in several window frames, it could be that a single object generated them (and they correspond to multiple split/merge events), or it could be two (or several) objects being very close to each other. It may seem that nothing was gained by the segmentation, but actually solving this problem is easier than before, because the search space is significantly reduced.

There are several ways that the segmentation problem can be solved. One way is to use a min-cut formulation similar to [19] or [5]. This, however, produces a "hard" segmentation without a confidence estimate, and restricts the form of the interaction between different detections. An alternative way that we pursue here is to determine the labeling in a generic probabilistic framework.

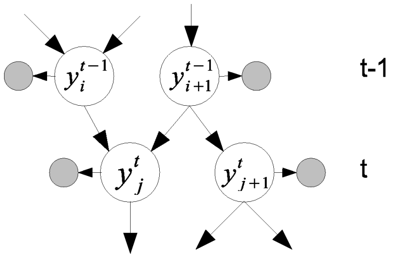


Figure 2: Example of the structure of the graphical model. Each y_i^t is a binary variable that represents a detection label. The shaded nodes represent the measurements associated with each detection (location, appearance, etc.).

3.2. Segmenting Detections

Let the label of each detection i at frame t be a binary random variable y_i^t, and let $\mathbf{y}$ denote all labels in the window. Let the observed properties of each detection (location, appearance, *etc.*) be denoted as $\mathbf{o}_i^t$, and let $\mathbf{o}$ denote all observations Then, the segmentation problem is

$$\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{o}) \ . \tag{1}$$

Solving this problem depends on how the joint distribution is formalized, allowing a great deal of flexibility. Here we factorize the joint distribution into a product of prior and conditional probabilities,

$$p(\mathbf{y}, \mathbf{o}) = p(y^0) \prod_{\substack{i,j,t>0 \\ y_i^t \text{near } y_j^{t-1}}} p(y_i^t|y_j^{t-1}) \prod_{i,t>0} p(\mathbf{o}_i^t|y_i^t) \ . \tag{2}$$

We let $y^0 = 1$ to denote the assumption that each detection in the first frame is valid. $\mathbf{o}_i^t$ denotes the observed properties of each detection (location, appearance, etc.). This factorization corresponds directly to the detection tree discussed earlier, and is illustrated in Figure 2.

One assumption that we are making here that may not be immediately apparent is that when a detection has multiple parent detections, independence of the parents is assumed. As a result, the conditional distribution conditioned on multiple parents is factorized into a product of simple conditional distributions, each conditioned on only one parent:

$$p(y_i^t|y_1^{t-1}, y_2^{t-1}, \cdots, y_K^{t-1}) \propto \prod_{k=1}^{K} p(y_i^t|y_k^{t-1}) \ . \tag{3}$$

This assumption is not correct when the parent detections come from one object (i.e. they are split), but it significantly simplifies the computation, and is not a big problem in practice. Without this assumption, the conditional probability table for a distribution conditioned on K parents would be 2 by 2^K.

Once the prior and conditional probabilities are specified, the solution to the segmentation problem is given by MAP inference. In addition, the max-marginals provide us with a confidence estimate of each detection. MAP inference is a well-studied problem, and can be solved using the max-product algorithm [10] or LP relaxation algorithms, such as [16]. Note that the structure of the factorization will have cycles in general, and as will be seen shortly, the conditional probability distributions, when viewed as energy functions, are not submodular, so the solution will be an approximation.

The conditional probability $p(\mathbf{o}_i^t|y_i^t)$ reflects the appearance similarity between the corresponding detection y_i^t and the initial detection y^0. Any appearance similarity measure can be used. It can be as simple as a sum of squared differences, or as complex as output of a classifier. For appearance similarity a that ranges in $[0, 1]$, the distribution is just

	$y_i^t = 0$	$y_i^t = 1$
$p(\mathbf{o}_i^t\|y_i^t)$	$1 - a(\mathbf{o}_i^t, \mathbf{o}^0)$	$a(\mathbf{o}_i^t, \mathbf{o}^0)$

The conditional probability $p(y_i^t|y_j^{t-1})$ is based on both the appearance similarity between the corresponding detections, as well as the motion likelihood of this detection given the preceding detections. The preceding detections are those which are on the path up to the root in the detection tree. There is a problem with this definition when a particular detection has multiple parents, because the motion model, which is described below, assumes only one observation at each timestep. To solve this problem we take the parent detection which gives the maximum motion likelihood, and call it the "motion parent." The effect of this is not to unfairly penalize valid detections that follow this ambiguity. For a motion likelihood m that ranges in $[0, 1]$, the conditional probability table is shown in Table 1.

This conditional distribution is a little bit complicated due to the asymmetry. The asymmetry is necessary, because we need a different behavior when the parent label is 0 (invalid detection) and when it is 1 (valid detection). When the parent detection is valid (bottom row), the distribution expresses that the probability is high when the detection label appears similar to the parent, and the motion likelihood after observing this detection is high (the motion is smooth). When the parent detection is invalid, however, we can not say anything about the appearance similarity. This is because a parent detection that is a false alarm (noise) may still have a similar appearance and motion. Therefore, in this case the distribution does not give a "hint" about the label of the detection.

	$y_i^t = 0$	$y_i^t = 1$
$y_j^{t-1} = 0$	0.5	0.5
$y_j^{t-1} = 1$	$1 - a(\mathbf{o}_i^t, \mathbf{o}_j^{t-1})m(\mathbf{o}_i^t)$	$a(\mathbf{o}_i^t, \mathbf{o}_j^{t-1})m(\mathbf{o}_i^t)$

Table 1: Conditional probability distribution used in the graphical model.

Any motion model can be used to determine the motion likelihood. Here we use a simple linear-Gaussian model:

$$\mathbf{z}_{t+1} = \mathbf{A}\mathbf{z}_t + \mathbf{w} \tag{4}$$

$$\mathbf{x}_{t+1} = \mathbf{H}\mathbf{x}_t + \mathbf{v} \tag{5}$$

where $\mathbf{z}_{t+1}$ is the state vector, which includes the object position and velocity, $\mathbf{x}_{t+1}$ is the measurement vector of the object position, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is the process noise, and $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is the measurement noise. The motion likelihood m is then

$$m(y_i^t) = \exp\left(-\frac{1}{2}\mathbf{e}_i^T\mathbf{P}_i^{-1}\mathbf{e}_i\right) \tag{6}$$

where $\mathbf{e}_i = \mathbf{z}_t^i - \bar{\mathbf{z}}_t^i$ and $\mathbf{P}_i = \mathbf{A}\bar{\mathbf{P}}_i\mathbf{A}^T$. $\bar{\mathbf{z}}_t^i$ is the posterior state estimate, $\mathbf{P}_i$ is the prior state covariance, and $\bar{\mathbf{P}}_i$ is the posterior state covariance in the previous time step. The model is initialized using the first two detections in the sequence.

Having specified the joint distribution, the MAP labeling is given by MAP inference. As indicated earlier, the valid detections in this labeling do not always define one tracklet. There may be extra detections due to noise, limitations of our model (a simplified factorization of the joint), the use of approximate inference, or when two objects with merged detections are splitting in this window. We describe how to handle this problem next.

3.3. Tracklets from detections

One way to solve this problem is to generate possible tracklets (hypotheses), and find zero or more that best explain the detections. Note that the search space is significantly reduced than before the segmentation, as we only need to explain the valid detections. Quite often there will be only one hypothesis.

The possible tracklets are generated by following the motion parent pointers up to the root of the tree from each valid detection, and removing any tracklet that is a prefix of another. See Figure 3 for an example. An elegant way to determine which (combination) of these best explains the detections is to set up a Quadratic Boolean Problem, as in [11]. We are investigating this approach, but here we present a reasonable, though admittedly ad-hoc, solution that works well.

First, we remove those tracklets that do not satisfy certain criteria. These criteria are:

- the number of detections in the temporal window must be at least half the window size

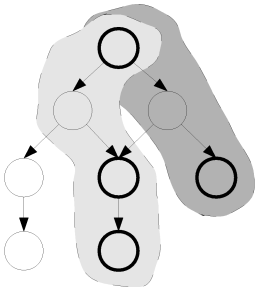


Figure 3: Generating possible tracklets from valid detections. Valid detections are denoted with thick outlines. In this case, two possible tracklets are generated as indicated by the shaded regions.

- the average acceleration of the object must be less than a threshold ($\approx 6\ m/s^2$)
- the object must be undergoing a smooth motion.

The average acceleration is estimated from the position of the detections, which are at least half a second apart. The elapsed time requirement is necessary to avoid bad estimates due to discretization noise. To determine motion smoothness for the third criterion, we compute the dot product of successive motion directions (at least half a second apart), transform it to $[0, 1]$ range, and compute the average. This average must be greater than a threshold (≈ 0.80) for the criterion to be satisfied. Note that this does not rule out tracklets that are making a turn, since the large change in direction will be filtered out by the average.

Another criterion that may be useful in aerial surveillance is to require that the object moves with some minimum speed, or travels a minimum distance. The effect of this is to filter out motion due to parallax. This is not applicable in all applications, however, so it is not included above. Other application-specific criteria can be defined.

The second step is to merge tracklets that are a result of split-merge events. This is done by repeatedly merging tracklets that have similar appearance and motion. More precisely, the similarity of two tracks τ_1 and τ_2 is computed using

$$sim(\tau_1, \tau_2) = \frac{1}{T}\sum_{1}^{T} sim(\tau_1, \tau_2, t) \tag{7}$$

$$sim(\tau_1, \tau_2, t) = a(\tau_1^t, \tau_2^t) * p(\mathbf{x}_1^t, \mathbf{x}_2^t) * v(\mathbf{v}_1^t, \mathbf{v}_2^t) \tag{8}$$

$$p(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|/c) \tag{9}$$

$$v(\mathbf{v}_1, \mathbf{v}_2) = \exp(-\|\mathbf{v}_1 - \mathbf{v}_2\|/c)\ , \tag{10}$$

where $a(\cdot)$ is the appearance similarity measure as before, $\mathbf{x}$ is the position of the detection, $p(\cdot)$ is a distance measure, $\mathbf{v}$ is the velocity at the time of the detection, $v(\cdot)$ is velocity similarity measure, and c is a constant to increase the dynamic range ($c \approx 32$). If the two tracklets share a detection in a particular frame, the similarity is automatically 1 for that frame, and if one tracklet has a detection in a frame, but the other one does not, $a(\cdot)$ is calculated using the previous detection, and $\mathbf{x}$, $\mathbf{v}$ are interpolated. When $sim(\tau_1, \tau_2)$ exceeds a threshold (≈ 0.90), the two tracks are merged.

3.4. Occlusion handling

So far in the discussion we have assumed there is no occlusion or missing data. It turns out that when an object is occluded but the occluder is detected, the algorithm as presented still works. This is because the detection tree does not really change, except that no detections in the frame where the object is occluded will be valid. A tracklet is still found, provided that the object is not occluded for most of the window. In that case, the tracklet would fail the first criterion above.

The real problem that needs to be handled is missing detections. When there is a missed detection, the detection tree will be shorter than T. If it is too short, any tracklets that are found will not have enough detections and fail the first criterion above. This problem is solved by adding "virtual detections" to the detection tree. These are added whenever a detection in frame t has no nearby detections in frame $t+1$. The position of this virtual detection is estimated using the motion model, and the appearance is copied from the (detected) parent. This procedure is recursive, so that when a newly added virtual detection does not have nearby detections in the next frame, the process is repeated.

4. Multi-object tracking

A multi-object tracker based on tracklets found by the presented algorithm works as shown in Figure 1. The step that we now address is the second one – associating tracklets with existing tracks. The task of this step is to form long tracks from tracklets found in the sliding window. Many strategies can be used here, but an important property to keep in mind is the many-many mapping. When two objects merge, and stay merged for a period longer than the window size, this association step must associate the one tracklet found with both existing tracks. Similarly, when one object has split detections for a period longer than the window size, the tracklets found need to be associated with the one existing track. Here we present one strategy that allows a many-many mapping.

We proceed again by defining an association similarity measure, resembling (7):

$$sim2(\tau_1, \tau_2) = \frac{1}{T}\sum_{1}^{T} sim2(\tau_1, \tau_2, t) \tag{11}$$

$$sim2(\tau_1, \tau_2, t) = a(\tau_1^t, \tau_2^t) * ps(\mathbf{x}_1^t, \mathbf{x}_2^t) \tag{12}$$

$$ps(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|/c)\ . \tag{13}$$

One difference here is that $\mathbf{x}$ is a vector containing the detection position as well as size. Also, if a track that we are comparing to does not have a detection at a particular frame (there must be at least one such frame since the sliding window shifted), the position of the detection is interpolated or extrapolated as necessary, and the appearance is taken from the most recent detection. This allows matching tracklets to tracks across long sensor gaps, assuming the motion of the existing track can be estimated accurately.

For a many-many mapping, we associate each tracklet with an existing track as long as $sim2$ is above a threshold. If we are only interested in matching each tracklet to one existing track, then we associate with a track having the maximum similarity, provided the similarity is above a threshold. In the latter case, a reasonable threshold is ≈ 0.40.

4.1. Implementation notes

We have implemented the algorithm just presented in C++. The implementation is available on the author's website. The runtime of the algorithm depends on the number of moving objects in the scene. When the number of objects is small (≈ 50), the runtime is around 1-2 frames per second on a 640x480 video. When the number of objects is in the hundreds, the runtime is several seconds per frame. Also, max-product was used for MAP inference.

5. Results

We have evaluated the multi-object tracker on sequences captured from an airborne sensor. The sequences come from the publicly available CLIF 2006 dataset [1]. The video is captured at roughly 2 Hz, and it is in grayscale. As this is a large format video roughly 6600x7500 pixels, we chose 640x480 subregions over an expressway for the purposes of evaluation. The sequences were stabilized prior to tracking. All of the sequences used in evaluation are available for download.

The only moving objects in the video are vehicles, but they are in very low resolution. Each vehicle is only about 7x7 pixels, which makes detection and tracking quite challenging. Since this low resolution gives very limited appearance information, we used a simple sum of squared differences function as our appearance similarity. This is computed after doing a least-squares alignment of the two image patches. The alignment is parameterized by translation and rotation. In this context, this parameterization is satisfactory.

The moving object detection was done using background subtraction. The background is modeled as the mode of a (stabilized) sliding window of frames. We have also tried a mixture of Gaussian model, but we did not see a significant difference. A window size of 16 frames, corresponding to about 8 seconds of video was used.

Ground truth tracks were manually generated for an 80-frame sequence containing 168 vehicles. Tracks shorter than the window size were not used in evaluation (for a window size of 16, this left 123 tracks). Several metrics were used to measure performance: object detection rate (ODR), false alarm rate (FAR), normalized track fragmentation (NTF), and ID consistency (IDC). The definition of the ODR and NTF is the same as in [13]. False alarm rate is the number of false detections divided by the number of total detections (NOT computed per track and averaged). The last metric is introduced to measure the tendency of the tracks to "jump" or switch IDs. To calculate this metric, we first label each detection in a ground truth track with the ID(s) of a test track(s), if any, which contains an overlapping detection. The ID consistency measures the largest fraction of detections in a ground truth track labeled with one label. Just as NTF, it is weighted by the length of the track to avoid favoring short tracks. More formally, let $G = \{g_i\}$ be the set of ground truth tracks, let $g_i = \{g_{ij}\}$ be the set of detections in each track, and $L(g_{ij})$ be the label of the test track associated with the detection g_{ij}. The ID consistency of a track g_i is

$$\text{IDC}_i = \max_l \frac{|\{g_{ij} \text{ s.t. } L(g_{ij}) = l\}|}{|g_i|} \tag{14}$$

and the overall ID consistency is

$$\text{IDC} = \frac{1}{\sum_{g_i} |g_i|} \sum_{g_i} |g_i| \text{IDC}_i \, . \tag{15}$$

The best ID consistency is 1.

Quantitative results using these metrics are shown in Tables 2 and 3. The first table shows the effect of using virtual detections in tracking. The second table shows the effect of different sliding window sizes on tracking. Qualitative results are shown in the following figures. Figure 4 shows tracking results on sample frames of the sequence used for evaluation. Figure 5 shows a specific example of handling merge-split events. Example of handling occlusion and sensor-gap is shown in Figure 6.

	ODR	FAR	NTF	IDC
With virtual det.	0.72	0.04	1.01	0.84
No virtual det.	0.61	0.03	1.04	0.90

Table 2: Effect of virtual detections on tracking performance.

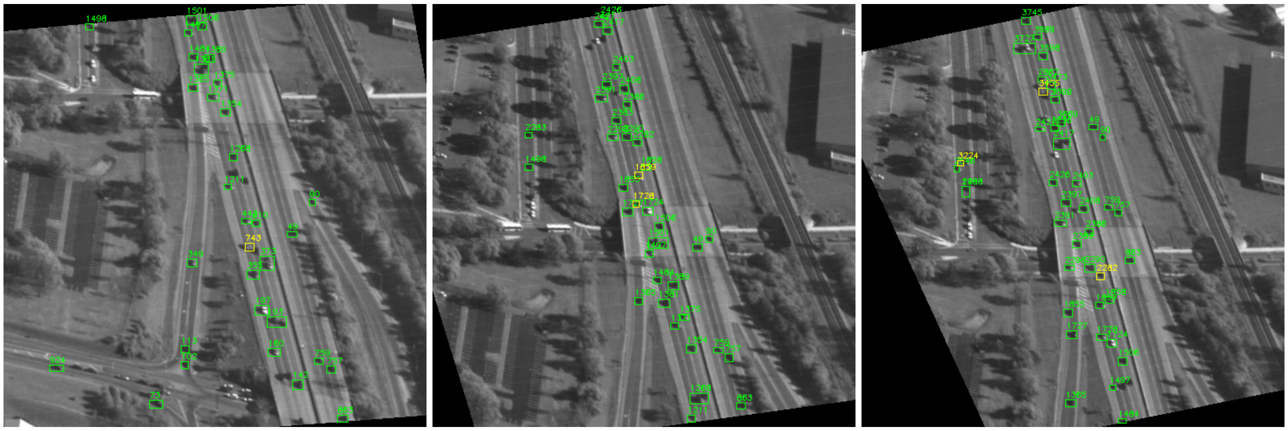
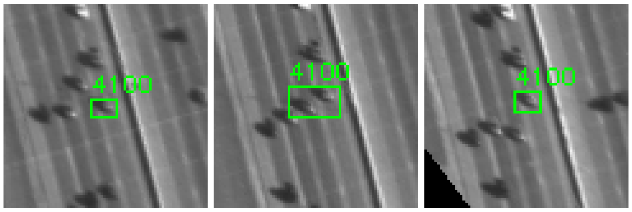
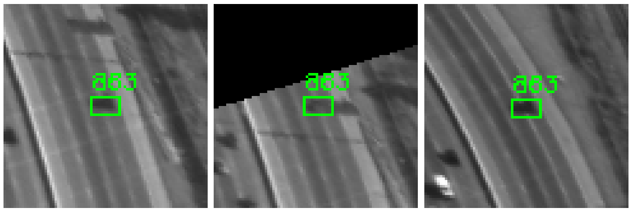


Figure 4: Tracking results on sequence 1. A green box denotes a real detection, whereas a yellow box denotes an interpolated detection.



(a) Frame 38 (b) Frame 46 (c) Frame 51

Figure 5: Example of merge event handling. Other tracks are not shown for clarity.

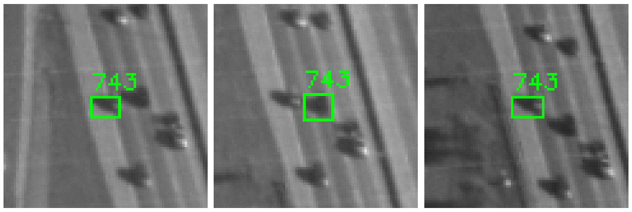


(a) Frame 49 (b) Frame 51 (c) Frame 54

Figure 6: Example of occlussion handling.

Window Sz.	ODR	FAR	NTF	IDC
10	0.76	0.04	1.06	0.86
12	0.73	0.04	1.00	0.85
14	0.72	0.04	1.00	0.87
16	0.72	0.04	1.01	0.84
18	0.71	0.05	1.00	0.83
20	0.63	0.04	1.00	0.89

Table 3: Effect of sliding window size on tracking performance.



(a) Frame 8 (b) Frame 10 (c) Frame 13

Figure 7: Example of a temporary ID switch. Other tracks are not shown for clarity.

5.1. Discussion

It is clear from the quantitative evaluation that the algorithm presented is very good at making detection associations. This is supported by the track fragmentation score, which is nearly perfect, as well as the ID consistency score. Once a track is established, tracking is unlikely to stop prematurely. The ID consistency score indicates that while it is possible that during tracking the ID may switch to another object, it is only a temporary change, as it never results in the original track being fragmented. An example of this can be seen in Figure 7.

Given the challenging nature of the video sequence, the object detection rate is quite good. Many of the detection failures are due to poor contrast between the object and the background. Another factor limiting the detection rate is the appearance similarity function. In our experiments we used sum of squared differences, which worked well, but there were cases where the alignment of two image patches did not converge correctly, and the computed similarity was low. However, thanks to the use of a sliding window and virtual detections, a missed detection in one frame does not cause the tracking to fail.

As a matter of fact, virtual detections are critical to achieving high object detection rate. Table 2 indicates that the object detection performance drops significantly when virtual detections are not used. It also shows that the false alarm rate slightly increases, however, it is more than compensated for the increased object detection rate.

The sliding window size mainly affects the object detection rate. As the sliding window size increases, more detections are necessary in order to declare a potential track valid. We would expect the false alarm rate to increase with decreasing window size, although it is not evident in the sequence used in evaluation. The false alarm rate is very low, despite some parallax motion due to buildings and trees in the scene.

6. Conclusions

Our primary contribution is a new algorithm for determining tracklets in a window of frames. This is a large data association problem between detections and an unknown number of objects, which we solve using inference in a set of Bayesian networks. Exhaustive evaluation of data association hypotheses is avoided, while allowing for split-merge conditions. The separation of valid detections from invalid detections is optimal in a MAP sense. In addition, we get a confidence estimate of each detection.

The algorithm was evaluated on a vehicle tracking task in an aerial surveillance video. The results show excellent performance in terms of object detection rate, track fragmentation, and false alarm rate on challenging video sequences.

We are currently working on a principled way to determine tracklets from valid detections using the Quadratic Boolean Problem. One avenue for future research is developing a more robust structure of the inference problem using higher-order relationships instead of using just pairwise. Another possible extension is to take advantage of other, application-specific, sources of information by incorporating them into the graphical model. For example, in the case of aerial surveillance, this can be a road network.

7. Acknowledgements

This work was supported in part by grant DE-FG52-08NA28775 from the U.S. Department of Energy. For the second author, this work was prepared by Lawrence Livermore National Lab under contract DE-AC52-07NA27344.

References

[1] CLIF 2006. https://www.sdms.afrl.af.mil/. 6

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on CVPR*, pages 1–8, 2008. 1

[3] I. Cox and S. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on PAMI*, 18(2):138–150, Feb 1996. 2

[4] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, 8(3):173–184, Jul 1983. 2

[5] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2):271–279, 1989. 3

[6] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision*, pages 788–801, 2008. 1, 2

[7] H. Jiang, S. Fels, and J. Little. A linear programming approach for multiple object tracking. In *IEEE Conference on CVPR*, pages 1–8, 2007. 2

[8] S. L. Junlian Xing, Haizhou Ai. Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *IEEE Conference on CVPR*, pages 1200–1207, 2009. 1, 2

[9] Z. Khan, T. Balch, and F. Dellaert. Multitarget tracking with split and merged measurements. In *IEEE Conference on CVPR*, volume 1, pages 605–610, 2005. 2

[10] F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, Feb 2001. 4

[11] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 878–885, 2005. 4

[12] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, volume 3021 of *LNCS*, pages 28–39, 2004. 2

[13] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *IEEE Conference on CVPR*, volume 1, pages 666–673, 2006. 1, 2, 6

[14] D. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843–854, Dec 1979. 2

[15] V. Reilly, H. Idrees, and M. Shah. Detection and tracking of large number of targets in wide area surveillance. In *ECCV*, volume 6313 of *LNCS*, pages 186–199. 2010. 1

[16] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening lp relaxations for map using message passing. 2008. 4

[17] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13, 2006. 1

[18] Q. Yu and G. Medioni. Multiple-target tracking by spatiotemporal monte carlo markov chain data association. *IEEE Trans. on PAMI*, 31(12):2196–2210, Dec. 2009. 2

[19] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *IEEE Conference on CVPR*, pages 1–8, 2008. 2, 3