Using 3D Scene Structure to Improve Tracking

Jan Prokaj and Gérard Medioni
University of Southern California
Los Angeles, CA 90089
{prokaj|medioni}@usc.edu

Abstract

In this work we consider the problem of tracking objects from a moving airborne platform in wide area surveillance through long occlusions and/or when their motion is unpredictable. The main idea is to take advantage of the known 3D scene structure to estimate a dynamic occlusion map, and to use the occlusion map to determine traffic entry and exit into these zones, which we call sources and sinks. Then the track linking problem is formulated as an alignment of sequences of tracks entering a sink and leaving a source. The sequence alignment problem is solved optimally and efficiently using dynamic programming.

We have evaluated our algorithm on a vehicle tracking task in wide area motion imagery and have shown that track fragmentation is significantly decreased and outperforms the Hungarian algorithm.

1. Introduction

There has been an intense effort in the Computer Vision community to solve the object tracking problem for many years. The goal of this effort is to accurately determine the location of moving objects in the scene at every timestep. Knowing this information is essential in applications such as surveillance, and human computer interaction, and provides the foundation for higher-level reasoning tasks, such as event and activity recognition.

As a result of this effort, the tracking performance has reached a level where we expect excellent results in many cases; when objects have rich appearance, for example [9]. Nevertheless, there still remain "corner" cases, which require a fundamentally different approach to be solved. One such case is when the moving object becomes occluded long enough for motion prediction to fail, or if the occlusion is short, but the object does not have a distinguishing appearance for successful reacquisition. These cases are difficult to solve without additional constraints. Yet, they occur often enough to warrant attention.

In this work, we focus on tracking vehicles from a moving airborne platform in wide area surveillance, where the kind of occlusions just described happens when vehicles move behind buildings and other structures. The appearance of vehicles here is very weak, usually only allowing us to distinguish light-colored vehicles from dark-colored vehicles, and small vehicles from large vehicles. Tracking vehicles across such occlusions is practically impossible without knowing more information. Here we explore the use of known 3D scene structure to estimate dynamic occlusion maps and improve tracking performance.

Figure 1. Automatically computed occlusion map (right) for the image on the left.

An occlusion map as used here is a binary map, which indicates what regions of the image are occluders of moving objects. See Figure 1 for an illustration. The occlusion map is not necessarily complete, there may be occluders which are not marked in the map (such as trees). Furthermore, the occlusion map may not be static for the duration of the image sequence due to camera motion. Importantly, the occlusion map is automatically computed from the video stream by estimating the camera pose in geo-coordinates and projecting a database of geo-referenced 3D models of buildings (occluders) to the image.

An occlusion map is useful, because it gives us the ability to detect occlusion events: a vehicle becoming occluded, a vehicle becoming visible. Analyzing these occlusion events, we can determine sources and sinks of traffic in the scene. While sources and sinks can be estimated without the use of an occlusion map [17], such techniques are not robust enough to be used in our domain, where many false

tracks arise due to the parallax motion of structures off the ground plane. By matching a sink with a source, we can apply an *ordering* constraint to solve the tracking problem: the sequence of vehicles becoming occluded should be approximately the same as the sequence of vehicles becoming visible. This sequence alignment problem is solved using dynamic programming. Note that by solving this problem we are able to correctly match vehicles before and after occlusion even if they stop or change motion while being occluded.

The primary contribution of this work is a novel use of 3D scene structure to improve tracking performance for objects moving through occlusions where motion prediction fails and/or when the appearance of objects is only weakly discriminative. We have evaluated the algorithm on real sequences from a publicly available wide area motion imagery dataset [2] and have shown that track fragmentation decreases and outperforms the Hungarian algorithm.

2. Related Work

The most related work to ours is the use of geo-spatial information to aid tracking [19, 20]. A recent example of this is work by Xiao *et al*. [19], where the camera pose is estimated in geo-coordinates to enable the assignment of semantic labels (building, tree) to image regions, and for road detection. The paper shows an increase in tracking accuracy when depth information, road network, and semantic scene segmentation is in use. However, there is only a a brief mention of using this information for occlusion handling in tracking, and no algorithm is given. In the followup work [20], an improved scalable tracker using GIS road network is presented, but again there is no mention of occlusion handling.

Tracking objects through occlusions has been considered most recently by Kaucic *et al*. [10] and Perera *et al*. [12]. In the earlier work of Kaucic, an occlusion map is estimated by segmenting the image into regions and training a classifier to label each region as one of a "building, road, grass, or vehicle." This occlusion map is then used to constrain location of image features for video stabilization, filter out false moving object detections, and detect when objects are likely to become occluded. One disadvantage of computing an occlusion map using image features is that it is generally not as accurate as one computed using known 3D scene geometry. Tracks before and after occlusion are linked using the Hungarian algorithm, where the pairwise cost is based on appearance similarity (color histogram matching) and motion prediction (constant velocity model).

Perera *et al*. extend this work by incorporating merge and split hypotheses to handle the many-many correspondence between moving region detections and objects. The Hungarian algorithm is again used as the computational engine for determining correct track associations. Template matching is used to define an appearance cost and a linear motion model to define a kinematic cost.

Besides the Hungarian algorithm, another class of methods for tracking occluded objects is based on a tracking context [5, 21, 8]. Our work also falls in this category as tracking context is implicitly formed when sequences of tracks are matched against other track sequences. In [5], the context is other moving objects, called predictors, whose motion correlates with that of the occluded object. The predictors are used to predict the location of the occluded object and associate it with the correct track upon reacquisition, and are found using Lyapunov Characteristic Exponent. A more sophisticated approach is presented in [21], where an object is continuously tracked collaboratively with auxiliary objects (the context), and the relationship between the auxiliary objects and the object of interest is modeled using a graphical model. Finally, most recently Grabner *et al*. presented an approach where the context of an object is a set of "supporters," which vote for the location of the object being tracked. As long as the supporters are visible, the location of the occluded object can be accurately determined.

The task of the driving algorithm in this work is to align two sequences of tracks. Sequence alignment is a well-known pattern recognition technique that has mostly seen its use in speech [14], and gesture (action) recognition [7, 6] to temporally align two sequences and in stereo correspondence to estimate disparities [15]. The classic method is Dynamic Time Warping (DTW), which is a dynamic programming algorithm to find the optimum non-linear alignment of two sequences. To our knowledge, sequence alignment has not been used before to track objects through occlusions.

3. Approach

The goal of our work is to improve tracking of moving objects, especially when they become occluded for a significant amount of time. We focus on the case where the occlusion is long enough to cause an object's track to break and become fragmented. The output of our algorithm is a track correspondence, which indicates what tracks should be merged or linked with other tracks.

As indicated in the introduction, we are concerned with tracking vehicles from a moving platform in wide area surveillance. Here, the appearance of vehicles is not very discriminative, and cannot be solely relied upon for correct track linking. Therefore, as has been done before, we combine appearance with a kinematic model, at least when the kinematic model is informative. Actually, the strength of our method is primarily evident when the kinematic model is unable to correctly predict a vehicle's location after occlusion.

We are able to accomplish this by applying an ordering constraint, which says that the sequence of vehicles becom-

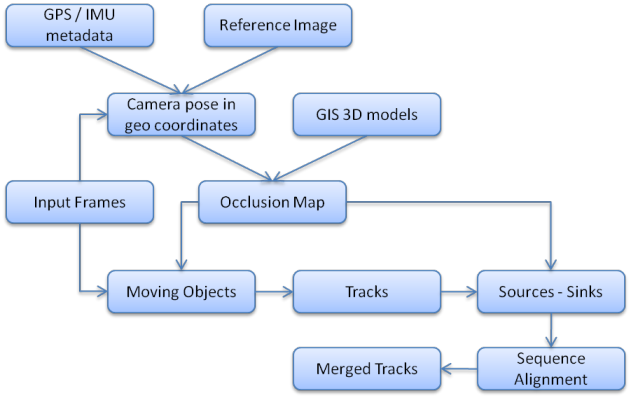

Figure 2. Our framework for tracking vehicles across long occlusions.

Figure 3. Example of the accuracy of the camera pose estimation. Accurate estimate of the camera pose results in a good registration of the 3D models (left) and occlusion map (right) with the image sequence.

ing occluded is approximately the same as the sequence of vehicles becoming visible after occlusion. In order to apply this constraint, we need to be able to reliably detect sources and sinks of traffic in the imagery and match them up correctly. A traffic sink is where vehicles become occluded and a traffic source is when they become visible again. Dynamic occlusion maps are essential for a robust estimation of these sources and sinks. The occlusion maps are computed automatically by estimating the camera pose in geo-coordinates and using a database of geo-referenced 3D models of buildings. These 3D models can be computed from imagery using dense 3D reconstruction techniques, or can be acquired from online sources [3] as we have done in this work.

A flowchart of our approach is in Figure 2.

3.1. Estimating the Dynamic Occlusion Map

The dynamic occlusion map is estimated by projecting a database of geo-referenced 3D models of buildings to the input video stream. Every pixel that is written during rasterization is marked as occluding. The occlusion map needs to be in agreement with the underlying imagery, but we do not require perfect registration. Accurate estimate of the camera pose in geo-coordinates enables us to achieve good registration.

GPS and IMU metadata associated with the input imagery provide a good initial estimate of the camera pose, but can not be used directly. To refine this initial estimate, we register the input imagery with a reference image (map). The input image transformation is modeled by a homography. We perform image registration in a fairly standard manner: by hierarchically estimating the image transformation, incrementally increasing the transformation complexity, using state of the art image features, and using a robust model estimator.

First, a Gaussian image pyramid is built for both the reference image and the video image. The images are first registered at the top of the pyramid, then the images at the next level down are warped using the just estimated transformation, registered again, and the process continues all the way down to the bottom of the pyramid which is at the highest resolution. This hierarchical process significantly increases the speed and accuracy of the registration, because at each level, feature correspondences only need to be found in a small local neighborhood.

The image transformation complexity is simple at the top of the pyramid and increases on the way down. More specifically, at the top levels of the pyramid, only image translation is estimated, and a full homography transformation is not estimated until the bottom levels. This is important, because at the top levels of the pyramid, the image resolution is not high enough to support a complex image transformation, so the estimation procedure is unstable.

State of the art image feature descriptors are used to establish correspondences at every level of the pyramid [18]. These are accurate and fast to compute. Finally, when estimating image transformation, RANSAC is used to provide a robust estimate. This is necessary to avoid using image correspondences on buildings and only make use of correspondences in planar areas.

Once we have a set of correspondences between the reference map and the video image that satisfy the final homography estimate, we solve for the camera pose by assuming all the correspondences lie on a plane. The algorithm presented by Schweighofer and Pinz [16] is used for this step. It provides a robust estimate of the pose by explicitly avoiding a solution that is a local minimum of the objective function.

The result of occlusion map estimation is illustrated in Figures 1 and 3. The average registration error is 10 pixels.

3.2. Tracking

Tracks of vehicles in the image sequence can be estimated by any multi-object tracking algorithm. By taking advantage of the just computed occlusion map, the number of false tracks can be significantly reduced. In our imple-

mentation, we used a tracker that computes tracks hierarchically – it first computes short tracks, and subsequently merges them into longer and longer tracks [13]. Computing tracks in this manner is much more robust than trying to compute long tracks from the beginning, since relying on smooth motion or consistent appearance over long time is unrealistic.

The main feature of the tracker we used is that the short tracks are computed optimally in a sliding window of frames. The problem of finding an optimal set of tracks is formulated by constructing a set of Bayesian networks, each of which models the joint distribution of detections over a fixed period of time. MAP inference in this graphical model then gives an efficient and optimal solution. Occlusions and missed detections are naturally handled in this model by assuming consistent object motion and appearance, which is a reasonable assumption over a short, fixed, time period.

3.3. Estimating sources and sinks

Our goal is to track vehicles even when they become occluded for a long time. Such long occlusions cause a tracking algorithm to lose a vehicle's track and initiate a new one when the vehicle becomes visible again. To achieve our goal, we need to find these false initiations and merge them with the correct earlier tracks that were prematurely terminated.

False initiations due to occlusion form a traffic "source" – a location where many tracks originate. Premature track terminations due to occlusion form a traffic "sink" – a location where many tracks stop. Of course, it is possible for a single vehicle to become occluded and reappear again. However, these locations may be difficult to identify with high confidence without other information, such as the road network. Fortunately, we can easily work around this problem by just waiting long enough for more tracks to enter the source and leave the sink.

The dynamic occlusion map is essential in detecting sources and sinks reliably. Source detection begins by taking all tracks that we have estimated (by the end of the sequence or a sliding temporal window), and computing the distance from the starting point of the track to the nearest occluder pixel. If this distance is "small", we record a leave-occlusion event at the track's starting location. The definition of "small" depends on the accuracy of the estimated occlusion map. In our work, we have used a value of 20 pixels (15 meters on the ground).

All the leave-occlusion events are then clustered in the (x, y, x', y') space, where (x, y) are the image coordinates of an occlusion event, and $x' = x + c \cdot v_x$, $y' = y + c \cdot v_y$, where v_x, v_y is the moving direction of the associated track. The constant c is used to give an appropriate scaling to the direction vector, and it is set to 20 in our experiments. Agglomerative clustering is used for this task: initially a cluster is created for each event, and the closest clusters are repeatedly merged until the distance between them reaches a maximum. The distance between two clusters is the Euclidean distance between the cluster centers. The only parameter of this clustering technique is the stopping criterion. In our work, we have used a value of 20 pixels. Note that due to the moving camera, the occlusion map is dynamic, and the occlusion events, while localized, will not be generated at a single point. The resulting clusters are the desired traffic sources.

Sink detection proceeds similarly, the only difference being that now we compute distance from the ending point of a track that has been declared terminated by the tracking algorithm. If this distance is small, we record an enter-occlusion event. After clustering these events we get the desired traffic sinks.

For efficiency, a distance transform is computed for an occlusion map. All the required distance calculations then only require a look up operation. To put this procedure into perspective, in a 801x233 sequence of 200 frames, we recorded 50 leave-occlusion events and 35 begin-occlusion events. After clustering, 4 sinks and 7 sources are produced. See Figure 4 for an illustration.

3.4. Source-Sink correspondence

After estimating sources and sinks, we need to know which sink corresponds with which source. Knowing this correspondence, we can then take the sequence of tracks entering the sink (becoming occluded) and align it with a sequence of tracks leaving the corresponding source (reappearing). The difficulty of this correspondence problem depends on the complexity of the occlusion. An occlusion where one sink is connected to more than one source (a one-many relationship), is considerably more difficult to handle than an occlusion with a one-one relationship between sink and a source.

In this work, we assume a one-one correspondence between sources and sinks, which is often satisfied in aerial surveillance imagery over urban areas. This does not mean that *all* vehicles entering a sink must leave a corresponding source. It is possible that a vehicle parks behind the occluding building, or more simply that a tracker does not track every vehicle entering / leaving the occlusion. However, we do expect that *most* vehicles entering a sink will in fact reappear at the source.

Using this assumption, we find a corresponding source for every sink by setting up a weighted bipartite graph matching problem, and solving it using the Hungarian algorithm. The pairwise matching similarity is defined as:

$$c_{ij} = C_1 * (d_1 + d_2) + (C_1/d) + C_2 * s(i) * s(j) \quad (1)$$

where d_1 is the dot product of the direction of traffic entering the sink with the direction of traffic leaving the source,

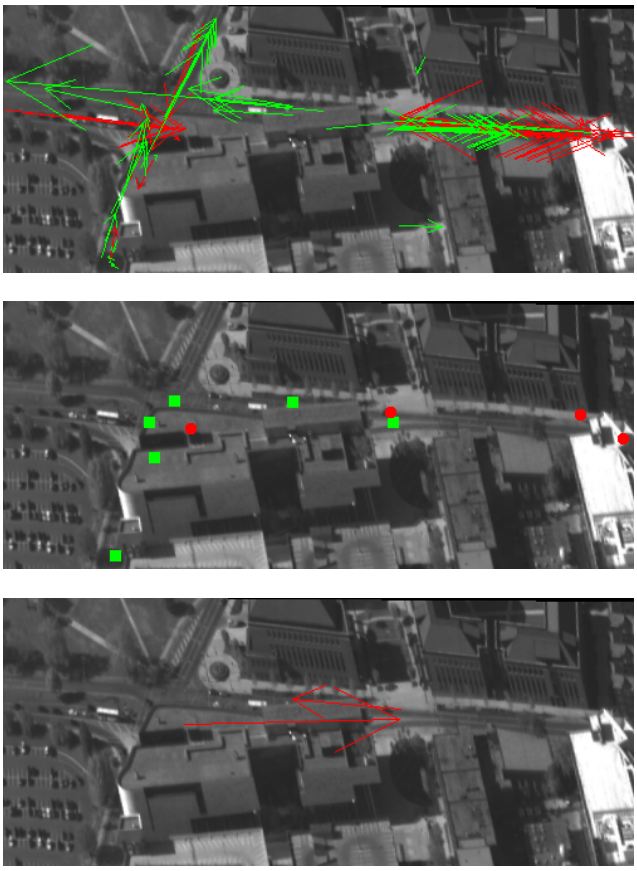

Figure 4. Example of source and sink detection. The top figure shows all occlusion events and the direction of travel for each one. The middle figure shows the resulting clusters – sources and sinks. Red circle indicates enter-occlusion and green square indicates leave-occlusion. The bottom figure shows correspondences between sources and sinks.

d_2 is the dot product of the direction of sink to source with the direction of traffic entering the sink, d is the distance from sink to source, $s(i)$ is the number of occlusion events (tracks) in cluster i, and C_1, C_2 are constants that weight each factor appropriately and make the matching similarity an integer. In our experiments, $C_1 = 1000, C_2 = 100$. The Hungarian algorithm will return an assignment of sinks to sources that maximizes this similarity. Furthermore, we require that the angle between the direction of traffic entering the sink and leaving the source is less than $60°$. A source-sink pair that does not satisfy this is removed from consideration. See Figure 4 for an illustration.

3.5. Sequence alignment

With an established correspondence between a sink and a source, our task is to match the sequence of tracks at the sink to the sequence of tracks at the source. By matching two sequences, and not two sets (as is customary with a Hungarian algorithm), of tracks we are applying an additional ordering constraint to solve the track linking problem. Using this additional constraint is essential when motion prediction fails (due to long occlusion) and/or when the appearance of the tracked objects is not very discriminative.

To solve the sequence alignment problem, we use a standard dynamic programming algorithm that is similar to Dynamic Time Warping and has been extensively used in computational biology. Let the sequence of tracks at the sink be $X = x_1 x_2 \cdots x_m$, and let the sequence of tracks at the source be $Y = y_1 y_2 \cdots y_n$. An alignment of the two sequences is a set of ordered pairs of tracks in X and Y such that there are no "crossing" pairs. In other words, if $(i, j), (i', j')$ is in our solution, and $i < i'$, then $j < j'$ [11]. This often holds in practice. To handle the case of a vehicle becoming out of order during the time it is occluded, we allow some tracks in X and Y to not match at all. For each such "gap", a penalty of δ is incurred. For every pair of tracks that do match, there is a matching cost $\xi_{x_i y_j}$. The cost of an alignment is the sum of its gap and matching costs. A dynamic programming algorithm minimizes this cost with the following recurrence [11]:

$$f(i,j) = \min[\xi_{x_i y_j} + f(i-1, j-1), \delta + f(i-1, j), \\ \delta + f(i, j-1)] \,. \tag{2}$$

In our domain, the matching costs ξ are computed from 4 terms, appearance term α, size term β, kinematic term γ, and feasibility term τ:

$$\xi_{x_i y_j} = \tau(x_i, y_j) * \gamma(x_i, y_j) * (\alpha(x_i, y_j) + \beta(x_i, y_j)) \tag{3}$$

The feasibility term is defined as

$$\tau_(x_i, y_j) = \begin{cases} 1 & \text{link } y_j \text{ with } x_i \text{ is feasible} \\ \infty & \text{otherwise} \end{cases} \tag{4}$$

where linking y_j with x_i is feasible if y_j becomes visible at a time after x_i becomes occluded, and given the distance and time between x_i and y_j, if the acceleration from x_i to y_j is physically possible ($\leq 10m/s^2$).

The kinematic term is defined as

$$\gamma(x_i, y_j) = \min(|t'_j - t_j|, G)/G \tag{5}$$

where t'_j is the predicted frame number of track x_i at the first location of y_j, and t_j is the actual frame number of track y_j at its first location. By clamping this time difference to G, the kinematic cost becomes unimportant for sequences of tracks which stop behind an occlusion, or otherwise have unpredictable motion. At the same time, if the motion prediction succeeds, the appearance and size terms become unimportant. The parameter G is set to 8, which corresponds to a time difference of 4 seconds in our imagery.

The appearance term is defined as

$$\alpha(x_i, y_j) = \min(D(h(x_i), h(y_j)), A)/A \quad (6)$$

$$D(p, q) = 0.5 * (D_{KL}(p||q) + D_{KL}(q||p)) \quad (7)$$

where D_{KL} is the Kullback-Leibler divergence, and $h(\cdot)$ is a probability distribution of image intensity on the foreground of the object. This distribution is calculated from a histogram of 16 bins. The parameter A is estimated as the maximum value of $D(p, q)$ in a training set of 11,208 appearance term computations. In our experiments, this value is 3.5.

The size term is defined as

$$\beta(x_i, y_j) = \min(|s(x_i) - s(y_j)|, B)/B \quad (8)$$

where $s(\cdot)$ is the median size of the tracked object. The parameter B is estimated as the maximum value of $|s(x_i) - s(y_j)|$ in a training set of 11,208 size term computations. In our experiments, this value is 17.

The parameter δ controls the likelihood of tracks becoming out of order. When it is 0, tracks before and after occlusion can be matched with an arbitrary number of gaps. When it is high, the first track entering an occlusion will always match the first feasible track leaving an occlusion. In our experiments, we have empirically determined the value of δ to be 0.40.

The alignment of tracks which minimizes the cost function $f(i, j)$ is the solution to our tracking problem. That is, by merging those tracks the vehicle continues to be tracked after the occlusion.

4. Computational Complexity and Implementation

The computational complexity of the sequence alignment algorithm is $O(nm)$, where n, m is the number of tracks in a sequence. In practice, this number is often small, so the algorithm is quite efficient. We have implemented the algorithm just presented in C++. The implementation is available on the author's website [1].

5. Results

We have evaluated the proposed track linking algorithm on sequences captured from an airborne sensor. The sequences come from the publicly available CLIF 2006 dataset [2]. This dataset is captured at roughly 2 Hz, and it is in grayscale. It is an example of persistent surveillance imagery, where the airplane makes several circular flyovers around the campus of Ohio State University.

Our database of geo-referenced 3D models has 90 models of buildings that were downloaded from Google Earth. For the geo-registration step of our camera pose estimation, we have used a reference image from USGS [4]. Camera calibration was achieved by manually selecting correspondences and performing bundle adjustment.

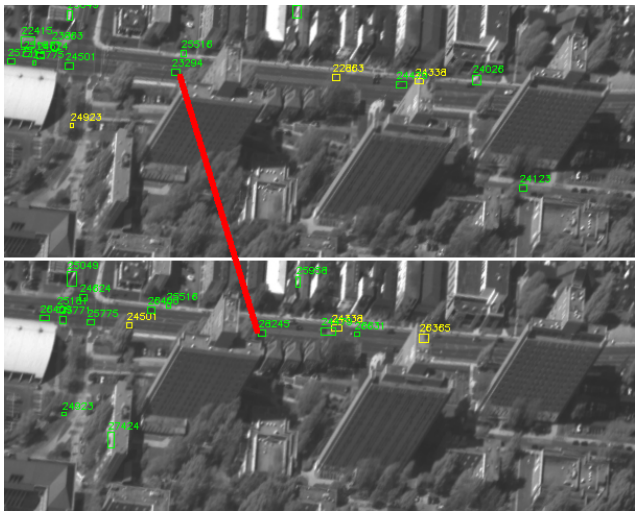

Figure 6. Correctly linked track by both the proposed algorithm and Hungarian algorithm. This is a short occlusion case, where motion prediction successful. This shows the algorithm works just as well in these cases. The top row shows the vehicle before occlusion and the bottom row shows the same vehicle after occlusion.

A baseline tracker [13] was used to estimate tracks in 5 sequences. The sequences are 150-200 frames long each, and roughly 500x200 pixels in size. We have selected those subregions of CLIF for which 3D models are available. In sequences 1 and 3, vehicles makes a stop while being occluded. Fragmented tracks in all sequences were manually identified in the tracking output. After running the proposed algorithm on each sequence, we manually verified the correctness of the track linking suggested by the algorithm. Ideally, all of the fragmented tracks identified earlier would be correctly merged together. The results can be seen in Table 1.

We compared the performance of the proposed sequence alignment to the widely used Hungarian algorithm. The matching cost used in the Hungarian algorithm was exactly the same as the one used in sequence alignment, but scaled and rounded to an integer value. The results of this comparison can also be seen in Table 1.

Examples of linked tracks in the sequences we tested can be seen in Figure 5, 6, 7, and 8. The test sequences, and other relevant data used in evaluation, are available on the author's website [1].

5.1. Discussion

The results show that the proposed algorithm is effective in linking tracks through occlusions. More than half of the fragmented tracks were successfully linked together. In contrast, the performance of the Hungarian algorithm is quite poor, with only a small number of tracks correctly

	Fragmented Tracks	Proposed Algorithm Correctly Linked	Proposed Algorithm Incorrectly Linked	Hungarian Algorithm Correctly Linked	Hungarian Algorithm Incorrectly Linked
Seq 1	9	5	2	2	7
Seq 2	4	3	3	3	4
Seq 3	3	1	6	1	7
Seq 4	2	1	2	1	3
Seq 5	6	4	1	3	3
Sum	24	14	14	10	24

Table 1. Quantitative evaluation of the proposed track linking algorithm.

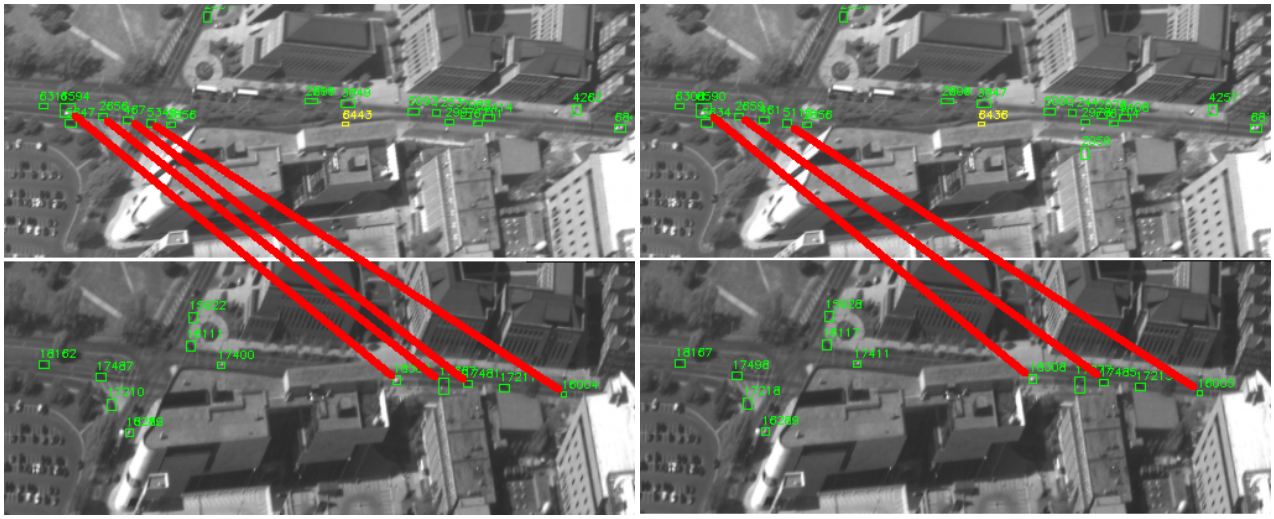

Figure 5. Linked tracks by the proposed algorithm (left) and Hungarian algorithm (right). Note that in this sequence, the vehicles stopped while being occluded. The only correct link by the Hungarian algorithm is the leftmost one. The top row shows vehicles before occlusion and the bottom row shows the same vehicles after occlusion.

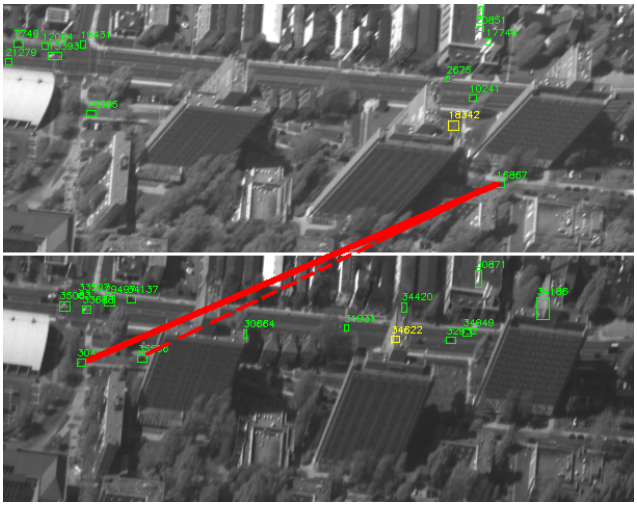

Figure 7. An example where both algorithms fail to link with the correct track, which is indicated with a dashed line. The top row shows vehicles before occlusion and the bottom row shows the same vehicles after occlusion.

Figure 8. Another example of a linked track by the proposed algorithm. The top row shows the vehicle before occlusion and the bottom row shows the same vehicle after occlusion.

linked and with a significant number of ID switches (incorrect links). Because we are using the same matching cost between tracks in both the sequence alignment problem and in weighted bipartite graph matching, we conclude that the ordering constraint is in fact responsible for the increased performance.

The most common type of error in the solution to the

sequence alignment problem is a track paired with a track that is next to the correct one (either ahead by one, or earlier by one). This can happen when the tracker does not track all vehicles entering or leaving an occlusion, and there is an appearance ambiguity as to which vehicle is the correct match. For example, when two dark colored cars are adjacent in a sequence, it is not clear how to choose the correct one. This does not happen when the appearance of vehicles is sufficiently diverse. In fact, the variation in appearance of vehicles acts like a synchronization mechanism – it prevents the error in one assignment from accumulating and throwing off the whole sequence.

6. Conclusions

Our primary contribution is a novel use of 3D scene structure to improve tracking performance for objects moving through long occlusions, and it is especially suitable when motion prediction fails and/or when the appearance of objects is only weakly discriminative. The key feature of the proposed solution is the application of an ordering constraint, which stipulates that the sequence of tracks becoming occluded is approximately the same as the sequence of tracks becoming visible. A dynamic programming algorithm is used to solve the sequence alignment problem.

The algorithm was evaluated on a vehicle tracking task in an aerial surveillance video. The results show excellent performance in terms of decreasing track fragmentation and outperform the Hungarian algorithm.

In the future we plan to consider the sequence alignment problem when multiple sources can be associated with one sink. We believe the knowledge of the road network is essential in this case.

7. Acknowledgments

This work was supported in part by grant DE-FG52-08NA28775 from the U.S. Department of Energy.

References

[1] http://iris.usc.edu/people/prokaj/. 1342

[2] CLIF 2006. https://www.sdms.afrl.af.mil/index.php?collection=clif2006. 1338, 1342

[3] Google Earth. http://www.google.com/earth/index.html. 1339

[4] USGS EarthExplorer. http://edcsns17.cr.usgs.gov/NewEarthExplorer/. 1342

[5] S. Ali, V. Reilly, and M. Shah. Motion and appearance contexts for tracking and re-acquiring targets in aerial videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 –6, 2007. 1338

[6] A. Bobick and A. Wilson. A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1325 –1337, Dec. 1997. 1338

[7] T. Darrell, I. Essa, and A. Pentland. Task-specific gesture analysis in real-time using interpolated views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(12):1236 –1242, Dec. 1996. 1338

[8] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1285 –1292, 2010. 1338

[9] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49 –56, 2010. 1337

[10] R. Kaucic, A. Amitha Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 990 – 997, 2005. 1338

[11] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson, 2006. 1341

[12] 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 Computer Vision and Pattern Recognition*, volume 1, pages 666 – 673, 2006. 1338

[13] J. Prokaj and G. Medioni. Inferring tracklets for multi-object tracking. Technical report, 2010. 1340, 1342

[14] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43 – 49, Feb. 1978. 1338

[15] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002. 1338

[16] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2024 –2030, 2006. 1339

[17] C. Stauffer. Estimating tracking sources and sinks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, volume 4, pages 35 –35, 2003. 1337

[18] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815 – 830, May 2010. 1339

[19] J. Xiao, H. Cheng, F. Han, and H. Sawhney. Geo-spatial aerial video processing for scene understanding and object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 –8, 2008. 1338

[20] J. Xiao, H. Cheng, H. Sawhney, and F. Han. Vehicle detection and tracking in wide field-of-view aerial video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 679 –684, 2010. 1338

[21] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1195 –1209, 2009. 1338