

Distributed multiple-hypothesis correlation and feedback with applications to video data

Kyle M. Tarplee, David J. Trawick, and Shawn M. Herman

Numerica Corporation, 4850 Hahns Peak Drive, Suite 200, Loveland, CO, USA

ABSTRACT

A common problem in video-based tracking of urban targets is occlusion due to buildings and vehicles. Fortunately, when multiple video sensors are present with enough geometric diversity, track breaks due to temporary occlusion can be substantially reduced by correlating and fusing source-level track data into system-level tracks. Furthermore, when operating in a communication-constrained environment, it is preferable to transmit track data rather than either raw video data or detection measurements. To avoid statistical correlation due to common prior information, tracklets can be formed from the source tracks prior to transmission to a central command node, which is then responsible for system track maintenance via correlation and fusion. To maximize the operational benefit of the system-level track picture, it should be distributed in an efficient manner to all platforms, especially the local trackers at the sensors. In this paper, we describe a centralized architecture for multi-sensor video tracking that uses tracklet-based feedback to maintain an accurate and complete track picture at all platforms. We will also use challenging synthetic video data to demonstrate that our architecture improves track completeness, enhances track continuity (in the presence of occlusions), and reduces track initiation time at the local trackers.

Keywords: Urban surveillance, video tracking, multi-sensor video data, centralized track fusion, tracklet feedback, single integrated track picture

1. INTRODUCTION

A common problem in video target tracking is the presence of obstacles, such as buildings or other structures, that can occlude objects of interest, such as people and vehicles, as they move through the coverage volume of a network of video sensors. This problem is compounded in congested urban environments, where it is also possible for targets of interest to occlude each other, such as when a bus obscures people on a nearby sidewalk. To improve tracker performance in the presence of line-of-sight obstacles, geometrically diverse cameras can be used to minimize the possibility of complete occlusion (i.e., relative to all sensors) within the coverage volume. However, because this sensor geometry provides overlapping coverage (by design), the video data from the sensors must be combined carefully, to avoid creating redundant tracks. In addition, the cost of communication between the video cameras and any track processing nodes must also be considered.

In this paper, we analyze three methods for combining the data from multiple video sensors into a common track picture. In all three cases, we assume that each video sensor is connected to a “local” tracker that generates source tracks (i.e., local tracks). However, the architectures that we will consider differ in how the local trackers share data with each other. The first architecture is referred to as *replicated-centralized measurement fusion*. In this architecture, each local tracker reports all its measurements to all other local trackers. If we ignored transmission delays, dropped messages, and frame processing order, the track pictures at each video tracker would be identical. Because all measurement data is distributed across all nodes, replicated-centralized measurement fusion establishes a performance baseline for the other architectures. However, it can require significant communication resources.

Further author information: (Send correspondence to K.M.T.)

K.M.T.: E-mail: kyle.tarplee@numerica.us, Telephone: (970) 461-2422 x231

D.J.T.: E-mail: david.trawick@numerica.us, Telephone: (970) 461-2422 x223

S.M.H.: E-mail: shawn.herman@numerica.us, Telephone: (970) 461-2422 x222

To reduce communication usage, tracklets can be transmitted between local trackers, instead of measurements. The tracklet-based approach provides a way to reduce bandwidth in a communication-constrained environment, albeit with a potential loss of accuracy for maneuvering targets. Because the dimension of a state vector (e.g., position and velocity) is typically larger than the dimension of a video measurement (i.e., line-of-sight to the target), the tracklet transmission rate should be significantly lower than the sensor measurement rate. Because of the significant delay between measurement generation and tracklet transmission, an explicit mechanism must be used to construct the integrated track picture, even if transmission delays are neglected. In a centralized architecture, a single node correlates the source track reports from all local trackers and creates an integrated track picture; in a distributed architecture, a local-to-remote correlation process occurs at each sensor node. The distributed architecture is more complicated because the track pictures at each node must remain synchronized, even in the presence of race conditions such as simultaneous track initiation on the same object at multiple nodes. For this paper, we have implemented the former scheme, which we refer to as *centralized track fusion*, in order to create a single integrated track picture with less communication usage than replicated-centralized measurement fusion.

Although the centralized track fusion architecture may achieve the objective of creating a single integrated track picture, this picture is only available at the central fusion node. The replicated-centralized measurement fusion architecture has the advantage of providing an integrated track picture at each local tracker. Fortunately, the integrated track picture in the centralized track fusion architecture can be distributed to the local trackers in an efficient manner using tracklet feedback. This third architecture, which we call *centralized track fusion with feedback*, provides an integrated track picture at each node without incurring the communication usage of replicated-centralized measurement fusion.

We begin by describing the three architectures in more detail in Section 2. Performance for each architecture will then be illustrated in Section 3 using synthetic video data from a challenging urban scenario involving vehicles, people, mounts and dismounts, temporary and persistent occlusion, and move-stop-move target motion. In the simulations, the scene is observed by video cameras aboard two Unmanned Aerial Vehicles (UAVs). We conclude in Section 4.

2. SYSTEM ARCHITECTURES AND ALGORITHMS

In this section we begin by describing our approach to video processing and video-based target tracking at the sensor nodes. These algorithms are common to all three network architectures that we will analyze. Then we describe the three centralized architectures proposed in the previous section for combining video data from multiple cameras into a single integrated track picture.

2.1 Video processing

All three architectures analyzed in this paper use the same set of algorithms for video processing. Each sensor platform contains one video processor composed of two parts. The first part performs motion-based segmentation, and the second part performs feature-based searching. Motion segmentation is accomplished using a *ray transform* algorithm. The ray transform utilizes an urban terrain map to alleviate registration problems due to parallax. In this algorithm, virtual lines of sight or “rays” are used to register several previous frames to the current frame for N -frame differencing. The ray transform can significantly reduce the number of false motion-based detections caused by edges of buildings and other forms of apparent motion when a scene is observed by a moving platform, such as a UAV.

A video tracker would be severely limited if it only received measurements from objects in motion. To address this issue, we supplement the motion detector with a method for finding stationary objects of interest in an image. This is accomplished by searching the image for regions that closely match the characteristics of existing tracks. More specifically, feature-aided tracking (see Section 2.2) provides estimates of color histogram and target length/width, which are used to search for a close match in the image. The search is limited based on each track’s estimated position and velocity. If a sufficiently good match is found, a feature-based measurement is added to the frame of motion-based measurements. These feature-based measurements will only have kinematic information (i.e., no features), to avoid biasing the subsequent track-to-measurement assignment problem.

2.2 Feature-aided video tracking

A Multiple Hypothesis Tracker (MHT)¹ that was developed as part of this research effort is being used for both the local measurement-based tracker and the centralized tracklet-based correlator. This MHT can process different types of observations (e.g., kinematic measurements, kinematic+feature measurements, and tracklets) and can produce different types of tracks (e.g., source tracks or system tracks). In brief, the MHT delays decisions about data association (i.e., the assignment of measurements to tracks) by maintaining multiple hypotheses over a window of several frames of data. This allows the tracker to consider competing hypotheses in regions of high data association ambiguity. Firm decisions are delayed so that future data can resolve the ambiguity.

At the sensor level, our MHT system has been extended to use video-specific features, such as object length/width and color histogram. All three architectures considered in this paper use the same filter to perform feature-aided tracking. Tracking estimates for target position and velocity are obtained using a hybrid central difference filter.² Length and width estimates are obtained using independent Kalman filters. The color histogram filter use a simple exponential forgetting scheme to update its estimates. The total track state consists of estimates of target position, velocity, length, width, and color histogram. A nearly-constant velocity model is used for the kinematic tracking, while the features use nearly-constant “position” models. Three-dimensional source tracks are constrained to the ground using terrain-based pseudo-measurements. The probability of detection for each track is computed by testing track visibility given the terrain map.

2.3 Centralized network architectures

2.3.1 Replicated-centralized measurement fusion

Figure 1(a) shows the measurement fusion architecture used in this paper. The measurements $\{z\}_i$, which include both kinematic and feature data, are sent directly to all UAVs. The local measurement trackers process the data from all sensors in the system. In general, using feature data from other platforms is questionable if the features are not invariant to the target/sensor aspect angle. For example, the length and width of a bus when viewed broadside are not the same as its length and width when viewed nose-on. Likewise, the color histogram of a person wearing a shirt with different colors on the front and back would vary with aspect angle. However, to give the local trackers the benefit of using feature data from remote platforms, in our simulations the two UAVs fly in a circular pattern and are diametrically opposed, so the length and width of each target is nearly the same for both cameras. Furthermore, the color of each person’s clothes do not vary from front to back. It is an objective of our future research to explore aspect-invariant feature models for multi-sensor tracking and fusion.

The replicated-centralized measurement fusion architecture implemented for this paper does not use Associated Measurement Reports (AMRs). This implies that the local tracker at each platform is free to assign the remote data that it receives as it sees fit. In general, this will lead to divergent track pictures at each node. However, because the measurement rate used in this work is relatively high (10 Hz) and the network delays are negligible, the track pictures at the local nodes will be nearly identical, even without AMRs. In general, this would not be true, though.

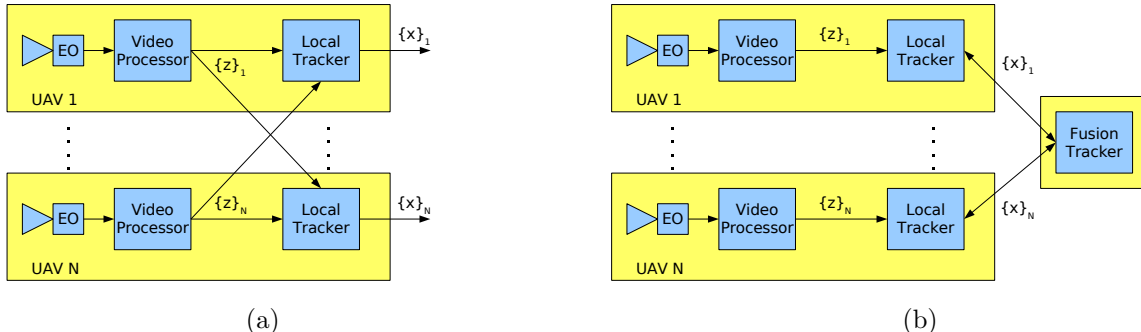


Figure 1. Multi-platform video tracking system with (a) replicated-centralized measurement fusion and (b) centralized track fusion with feedback.

2.3.2 Centralized track fusion

Although a wide variety of architectures are available for track fusion systems,³⁻⁵ in this paper we explore a centralized approach where the data from all local trackers are fused into system-level tracks at one location. A block diagram of this configuration is shown in Figure 1(b). The path for track feedback from the fusion node to the local trackers is also shown, but it is only part of the architecture discussed in Section 2.3.3.

In this paper we use tracklets to convey track information between the local trackers and the fusion node. The tracklets are obtained by the *local restart* technique. More specifically, the local tracker “remembers” which measurements have been part of the tracklets previously sent to the fusion tracker. At each reporting time, the local tracker prepares tracklets to send to the fusion node by initializing each source track and refiltering using its unreported measurements (i.e., the measurements associated with the track since the last reporting time). In this way, the system tracks at the fusion node are only updated with measurements that have not previously contributed to them. The system tracks are updated using the tracklets in a manner similar to measurement fusion, with the exception that the tracklets contain velocity estimates. An advantage of locally-restarted tracklets versus the Frenkel method⁶ is that they do not require covariance subtraction, which can lead to ill-conditioned matrices. However, the disadvantage of local restart is the computational cost of refiltering, although this cost is small when compared to the required video processing.

Using locally-restarted tracklets does not remove common process noise. Although the source tracklets are created from measurements with independent errors, the common motion models used by the local trackers lead to common process noise. For example, if all the local trackers were using a nearly constant velocity model for filtering, and the target suddenly turned left or right, the subsequent estimation errors across the local trackers would be correlated, which would violate the Kalman filter assumptions. This problem can be mitigated by using the Best Linear Unbiased Estimator (BLUE) filter⁷ or by allowing the process noise to increase, if applicable.

The outgoing messages from the local trackers contain two types of information: locally-restarted source tracklets and coasted track IDs. The tracklets are created according to the associations in the root of the MHT hypothesis tree because all of its assignments are “firm” (i.e., cannot be changed). For this reason, the data sent to the fusion node will be delayed by the time span of the MHT window. In order to form a source tracklet, there must be a sufficient number of measurements in the reporting period to yield a reasonable state estimate. Source tracks that do not have enough measurements for tracklet formation are reported to the fusion node without a state, in the coasted track ID list. This scheme conveys the complete local track picture to the fusion tracker with a minimal amount of data.

2.3.3 Centralized track fusion with feedback

The centralized track fusion with feedback architecture is very similar to the previous architecture. Its design was motivated by the desire to have faster local track initiation and the ability to maintain source tracks through occlusions at each sensor node. Both of these design goals can be achieved if the fusion node’s track picture is distributed to all local nodes. Thus, the fusion tracker’s primary feedback responsibility is to send track data that will produce the same track picture at each local tracker. As discussed before, at each periodic update time the locally-restarted tracklets and the coasted track ID list are sent to the fusion node. The local tracker then waits for a response in the form of feedback before processing any more local measurement frames. In this paper, the local trackers interact with the fusion node in an interleaved fashion, to reduce spikes in communication and improve the performance of the system tracker.

Once the system tracker finishes processing the source tracklet frame, the best hypothesis is used for feedback. The fusion tracker transmits source tracklets from other UAVs instead of the system tracks themselves, to avoid sending redundant data to the local trackers. Two types of tracklets are fed back to the UAVs: *new* tracklets and *improved* tracklets. New tracklets are created from system tracks for which there is no correlation to any of the local node’s tracks. These new tracklets will initiate source tracks in the local tracker, in order to maintain a complete track picture. An improved tracklet is sent when the size (determinant or trace) of the predicted covariance from another UAV’s source tracklet is smaller than the size of the covariance from the current UAV’s tracklet for the same system track. This can happen when the local tracker stops receiving measurements on an occluded object. In this way, tracking through occlusions is made possible by using the data from other UAVs in the network.

3. SIMULATIONS AND MONTE CARLO ANALYSIS

3.1 Simulation environment

ALTO, which is Numerica’s MATLAB[®] simulation environment, was used to test prototype versions of all three architectures described in Section 2.3. ALTO is a discrete event simulation system that supports rapid algorithm development. The network architecture of the simulated system is specified using a collection of configuration files that are easy to adapt, and the input parameters are specified through textual input files. ALTO is also designed to facilitate multiple Monte Carlo runs of a given scenario.

We used simulated video data as the input for our Monte Carlo tests. A commercially-available modeling, animation, and rendering program called Maya^{®8} was used to generate high-fidelity video data based on the true trajectories of the targets in the scene. The truth trajectories were also available for evaluation of each architecture’s performance using ALTO’s built-in metrics package.

3.2 Scenario description



Figure 2. (a) View from UAV1; (b) view from UAV2. The local track estimates are displayed as position covariance ellipses (green or red) and velocity covariance ellipses (blue). Note that two tracks of UAV2 are currently occluded and are not receiving local measurement updates (signified by the red color of their position ellipses), resulting in significantly larger covariance estimates. Nevertheless, these tracks are maintained by track feedback from UAV1 (via the fusion node).

The scenario consists of two UAVs, both flying counter-clockwise (looking down) at constant altitude on opposite sides of the same circle undergoing turbulence. Both are observing the same urban environment where ten targets of interest are present, including seven people and three automobiles. An example of the views of both UAVs at the same time is given in Figure 2. The targets are frequently occluded by the buildings and other targets in the scenario. Some of the people also enter and exit buildings and cars. The people are modeled with articulated motion in the torso, arms, and legs.

Figure 3 shows an overhead view of the terrain (to scale) marked with the trajectories of the targets of interest. The three cars in the scene are labeled by type: Mini Cooper, Pickup, and Datsun. The seven people in the scene are labeled as Person 1 to Person 7. The label for each target is placed at the beginning of its trajectory. A chronology of the scenario’s special events, such as target maneuvers, interactions, and occlusions, are enumerated in Table 1. Table 2 gives the metrics scoring start and stop times for each of the targets.

Table 1. Sequence of Events

Event	Time (s)	Description
A	0.0	Person 1, Person 2, and Person 5 are in the scene.
B	8.3	Pickup enters the scene from the east.
C	12.8	Person 5 follows Person 1 closely while both are turning.
D	17.1	Pickup stops.
E	21.2	Person 4 and Person 6 enter the scene from the east.
F	22.3	Person 7 enters the scene from the west.
G	22.9	Person 3 dismounts the Pickup, and the Pickup starts moving again.
H	30.0	Datsun starts moving.
I	31.5	Pickup partially occludes Person 1 as it drives by.
J	34.3	Datsun starts to turn south.
K	37.0	Person 5 enters a building.
L	37.8	Person 1 mounts the Pickup after it has stopped.
M	39.7	Person 3 enters a building.
N	42.3	Mini Cooper enters the scene from the east.
O	46.1	Person 6 enters a building.
P	46.6	Pickup starts moving again.
Q	49.0	Person 4 and Person 7 collide and reverse directions.
R	57.1	Pickup exits the scene to the west.
S	60.0	Scenario ends.

Table 2. Active Truth Target Times for Metrics

Truth Target	Start Time (s)	Stop Time (s)
Mini Cooper	39.5	58.5
Pickup	7.5	56.5
Datsun	24.5	40.5
Person 1	1.5	37.15
Person 2	1.5	13.5
Person 3	23.5	38.5
Person 4	21.5	60.0
Person 5	1.5	35.5
Person 6	20.5	45.5
Person 7	23.5	60.0

Table 3. Communication Load and RMS Position Error

Network Architecture	UAV1			UAV2		
	Comm. Load with Features (bytes)	Comm. Load without Features (bytes)	RMS Pos. Error (m)	Comm. Load with Features (bytes)	Comm. Load without Features (bytes)	RMS Pos. Error (m)
Measurement Fusion	5,850,780.8	153,464.0	0.267	4,443,960.0	127,020.0	0.284
Track Fusion, no FB	284,240.0	28,042.4	0.385	213,236.8	21,140.8	0.529
Track Fusion, FB	298,001.6	30,737.6	0.392	290,923.2	29,923.2	0.468

(across Monte Carlo runs) total communication load for each architecture assuming 8-byte numbers.

A major distinction between the three architectures is their communication load. As expected, the communication load for the measurement fusion case is considerably larger than either of the track fusion cases, while the communication load of the track fusion case with feedback is only somewhat larger than the case without feedback. The relative advantage of the track fusion architectures over measurement fusion is greater when feature data is sent over the network. This is because the storage required for our feature states (primarily the color histogram) is much larger than storage for the kinematic track state.

The RMS errors of the track position estimates, computed using the true positions of the targets, are also reported in Table 3. These results show that, even though there is a great reduction in communication load by using track fusion instead of measurement fusion, there is not much of an increase in the RMS position error. Note that the RMS errors in Table 3 are for the local trackers (which are using features in our experiments). The RMS errors for the system/fusion trackers, for the feedback and no-feedback cases, are 0.656 m and 0.662 m respectively. The system track position estimates are not as accurate as those of the local trackers because the update rate of the fusion node is ten times slower.

We now evaluate the integrity of the track picture at each local tracker (i.e., UAV) for our three architectures. The integrity of the track picture can be characterized using the following metrics: tracker completeness, track initiation time, track breaks and switches, redundant track ratio, and spurious track ratio. We have computed each of these metrics over the same Monte Carlo runs by assigning local tracks to truth objects using a distance-based cost. In the following figures (except the bar plot for track initiation time), performance of replicated-centralized measurement fusion is shown using a red dashed line, performance of centralized track fusion (“no FB”) is shown using a solid green line, and performance of centralized track fusion with feedback (“FB”) is shown using a solid blue line.

The tracker completeness metric is shown in Figure 4 for each network architecture. Tracker completeness is computed as the number of associated tracks divided by the number of truth objects. The results for the three architectures are similar for UAV1 because most of the targets are in its field of view (FOV) throughout the scenario. However, the situation is different for UAV2 because several of the targets are occluded for significant portions of the scenario. Thus, the completeness of track fusion without feedback is degraded relative to the measurement fusion case. Track fusion with feedback, however, essentially restores all the missing tracks to the local track picture of UAV2, yielding comparable performance to measurement fusion.

The track initiation times of the three methods are shown in Figure 5. Track initiation time is computed as the difference between the time that an object enters the scene (i.e., FOV of any sensor) and the time that it begins to be tracked. As with tracker completeness, the track initiation times for the three architectures are similar for UAV1. However, several track initiations at UAV2 are delayed in the track fusion without feedback architecture because of occlusion. Furthermore, a track on Person 2 is never begun (Person 2 is always occluded), which causes the track initiation time to equal the scenario duration. The initiation times for the measurement fusion case and the track fusion with feedback case are similar for all targets.

There are very few track breaks with any of the architectures, as seen in Figure 6. A track break occurs when a truth object, which was previously associated with a track, is no longer associated with any track. The

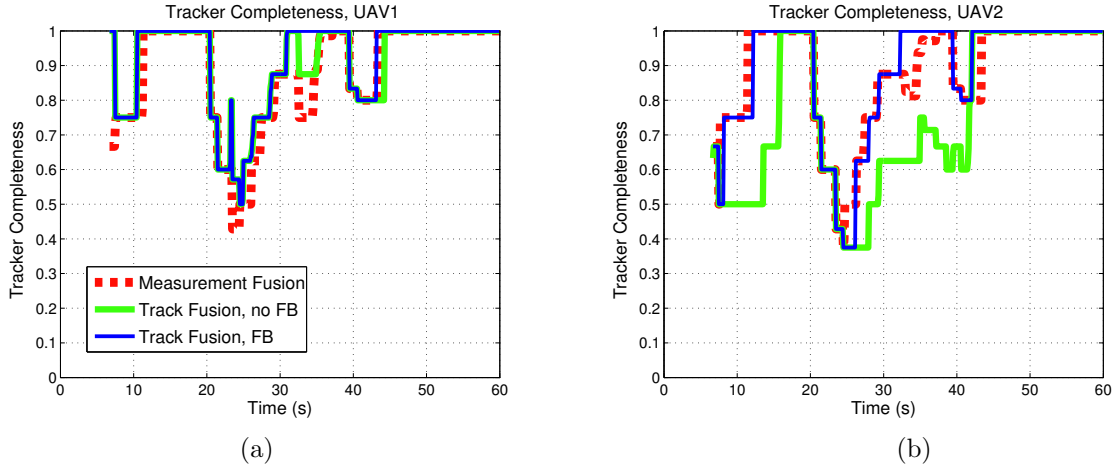


Figure 4. Tracker completeness for (a) UAV1 and (b) UAV2.

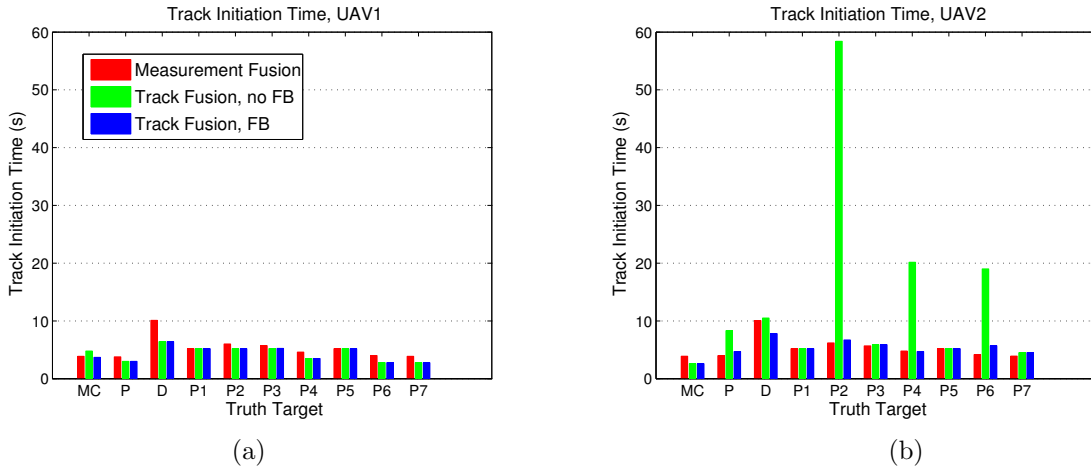


Figure 5. Track initiation times for (a) UAV1 and (b) UAV2. ‘MC’ refers to the Mini Cooper, ‘P’ refers to the Pickup, ‘D’ refers to the Datsun, and ‘P n ’ refers to Person n .

mean cumulative number of track breaks is the total accumulation of track breaks divided by the number of Monte Carlo runs times the number of targets (i.e., $10 \times 10 = 100$). Therefore, a track break that occurs on the same target in all ten Monte Carlo runs will produce a jump of 0.1 in the track break plot. The track break that occurs at 31.5 s corresponds to Event I of Table 1, where Person 1 is partially occluded by the Pickup as it drives by. Because UAV2 has a better view of this interaction, it is able to maintain track on the person. Thus, with track fusion and feedback, both local trackers are able to avoid this break. However, measurement fusion does not perform as well in this specific instance. It is likely that the track on Person 1 is dropped because every other measurement (i.e., those from UAV1) are misses during the partial occlusion. Furthermore, the score of the track on Person 1 for measurement fusion appears to be on the verge of the drop threshold because the slightly different order of sensor frame processing at the UAVs results in fewer track breaks at UAV2 than UAV1.

Figure 7 shows the mean cumulative number of track switches. A track switch occurs when a truth object is associated to a different track at the current scoring time than it was at the previous scoring time. Similar to track breaks, this metric is normalized by the number of Monte Carlo runs times the number of targets (i.e., 100). There are several track switches near 50 s, which corresponds to Event Q where Persons 4 and 7 collide. An analysis of movies generated with the local tracks overlaid on the input images (e.g., Figure 2) for all Monte Carlo runs did not reveal any actual track switches. Because ALTO’s metrics use the distance between tracks

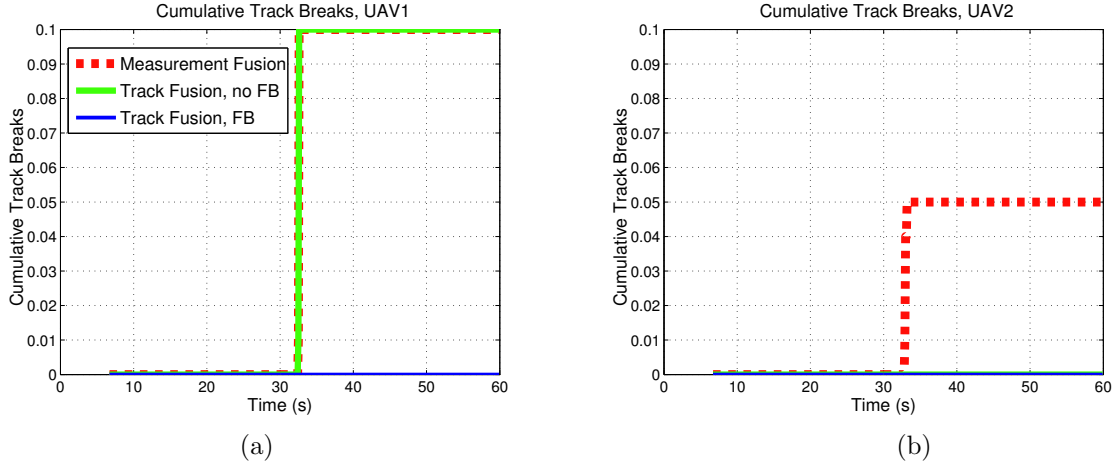


Figure 6. Cumulative track breaks for (a) UAV1 and (b) UAV2.

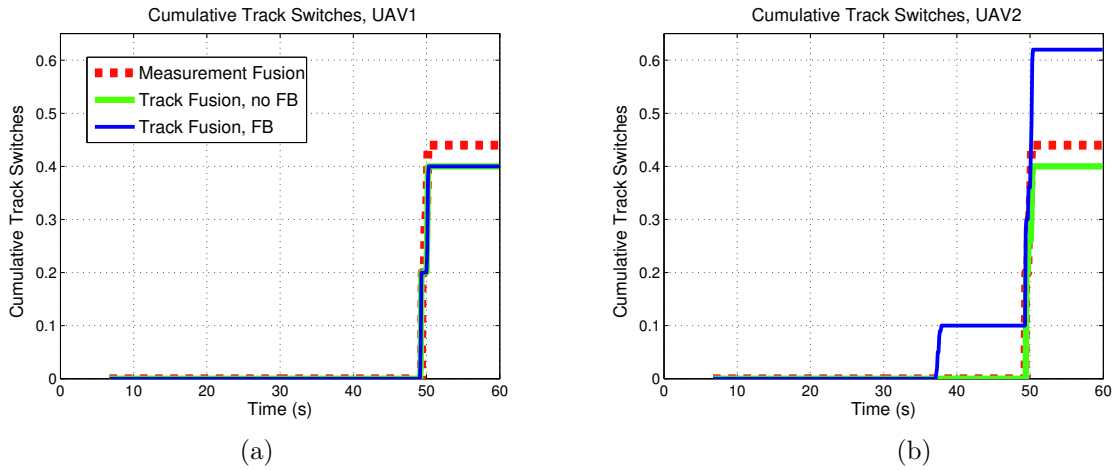


Figure 7. Cumulative track switches for (a) UAV1 and (b) UAV2.

and truth objects to find the globally optimal track-to-truth assignment, when tracks are proximate the metrics correlator can swap its associations. During Event Q, the truth objects are very close to each other (less than 50 cm separation). However, because our local trackers use features in addition to kinematic data, they are able to maintain tracks on both people during Event Q without switching (even though their position estimates may indicate that a switch has occurred). The track fusion with feedback case for UAV2 shows an additional switch near 38 s, corresponding to Event L, when Person 1 mounts the Pickup. This is also caused by confusion in the metrics correlator when the tracks on these targets become close to each other.

Proper track termination has not been added to the MHT used for these experiments, which causes tracks to persist longer than they should when targets enter buildings or vehicles. A track is dropped once its cost becomes poor, but this can take 2–3 seconds to occur after a target enters a building, mounts a car, or leaves the fields of view of both UAVs. Inefficient track termination is primarily responsible for the redundant and spurious tracks of our three network architectures.

Figure 8 shows the redundant track ratio. The redundant track ratio is the number of tracks that gated with at least one truth object divided by the number of tracks that were assigned (by the metric) to a truth object. Therefore, the redundant track ratio is always greater than or equal to 1.0, with 1.0 being the optimal value. The redundant tracks in Figure 8 are caused by improper track termination. During the spike around 40 s, the

track on Person 5 is drawn toward the nearby Datsun by spurious measurements after Person 5 enters a building (Event K). Similarly, the track on Person 1 is confused with the Pickup after Person 1 mounts it (Event L). The spike for UAV2 at about 32 s is caused by a misassociation in the metrics routine when the Pickup passes behind Person 1 (Event I). The plots show that these redundant tracks do not last more than a couple seconds, and the overall behavior is very similar, regardless of network architecture. However, the redundant track ratio is higher for UAV2 in the track fusion without feedback case because it does not track as many objects (i.e., the denominator of the ratio is smaller).

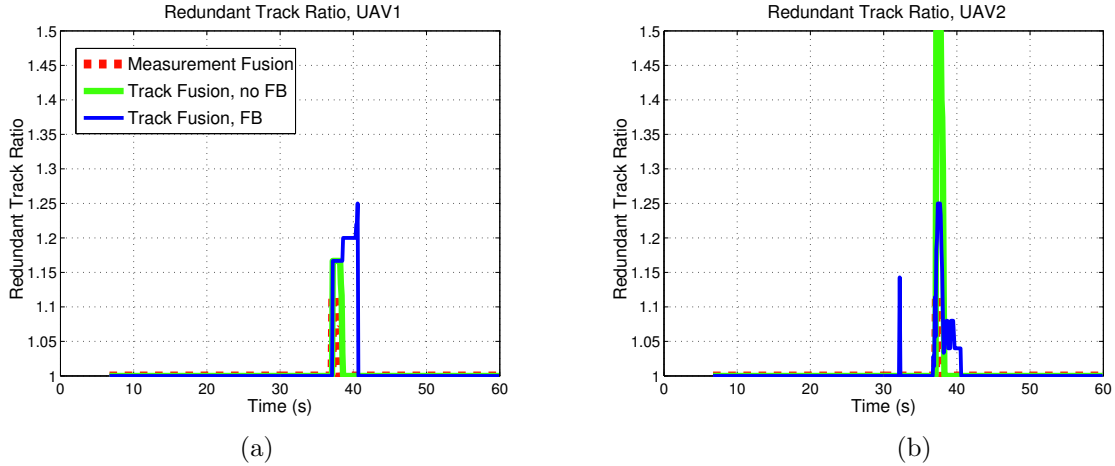


Figure 8. Redundant track ratio for (a) UAV1 and (b) UAV2.

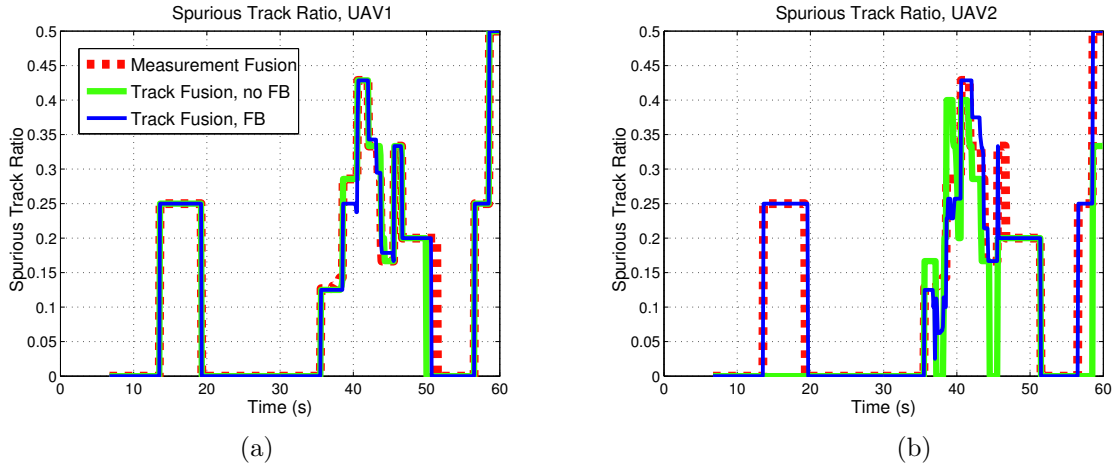


Figure 9. Spurious track ratio for (a) UAV1 and (b) UAV2.

Figure 9 shows the spurious track ratio. This ratio is computed as the number of tracks that did not gate with any truth object divided by the number of tracks at that time. Therefore, the spurious track ratio is always between zero and one. As mentioned previously, the only spurious tracks we have are those that are not terminated promptly when their corresponding truth objects leave the scene. ALTO’s metrics routine will label these tracks as “spurious.” For example, the spurious track ratio jumps to 0.25 between 14 s and 20 s because Person 2 leaves the scene, but there is a delay before its track is dropped by UAV1’s local tracker. During this time, UAV1 has four tracks, but truth objects only exist for three of them, yielding a spurious track ratio of $(4 - 3)/4 = 0.25$. The many spikes between 35 s and 52 s are caused by people exiting the scene by entering

buildings. The spike at the end of the scenario is caused by the Pickup and Mini Cooper leaving the scene. The spurious track plots are very similar, regardless of network architecture.

4. CONCLUSIONS

In this paper, we have considered the challenges of target tracking in an urban environment, particularly object occlusion. We showed that tracking performance can be improved by fusing data from geometrically diverse video sensors. We also presented three centralized fusion architectures along with engineering trade-offs associated with each. We then evaluated the three architectures using synthetic video data from a challenging urban scenario involving vehicles, people, mounts and dismounts, temporary and persistent occlusion, and move-stop-move target motion. Our analysis showed that centralized track fusion with feedback has performance that is comparable to replicated-centralized measurement fusion, but without the large communication load. Furthermore, both track fusion with feedback and replicated-centralized measurement fusion are able to maintain an integrated track picture at each sensor node. Conversely, although the communication load of track fusion without feedback is lower than track fusion with feedback, it cannot provide a common track picture at all nodes. Future work will explore decentralized network architectures for multi-sensor video-based target tracking.

ACKNOWLEDGMENTS

This work was supported by the Air Force Research Laboratory (AFRL) under contract FA8650-05-C-1805. We would also like to gratefully acknowledge the guidance of Dr. Jeffery Layne and R. Alan Wood of AFRL.

REFERENCES

1. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, Norwood, MA, 1999.
2. M. Nørgaard, N. K. Poulsen, and O. Ravn, “Advances in derivative-free state estimation for nonlinear systems.”
3. S. Lu and A. B. Poore, “Network-centric MFA architectures,” in *Cooperative Control: Models, Applications and Algorithms*, P. Pardalos, R. Murphy, and S. Butenko, eds., pp. 187–213, Kluwer, 2002.
4. R. Novoselov, S. Gadaleta, and A. Poore, “Network-centric MFA tracking architecture based on soft-level data association,” in *Proceedings of the SPIE, Signal and Data Processing of Small Targets*, **5913**, 2005.
5. S. Lu, A. B. Poore, and B. Suchomel, “Network MFA tracking architectures,” in *Proceedings of the SPIE, Signal and Data Processing of Small Targets*, **4473**, pp. 447–457, 2001.
6. G. Frenkel, “Multisensor tracking of ballistic targets,” in *Proceedings of the SPIE, Signal and Data Processing of Small Targets*, **2561**, pp. 337–346, 1995.
7. Y. Zhu and X. R. Li, “Best linear unbiased estimation fusion,” in *Proceedings of the 2nd Annual International Conference on Information Fusion*, pp. 1054–1061, 1999.
8. Autodesk Maya. <http://www.autodesk.com/maya>.