Event Recognition in Airborne Motion Imagery

J. B. Burns, C. I. Connolly, J. F. Thomere, M. J. Wolverton

Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA
burns@ai.sri.com, connolly@ai.sri.com, thomere@ai.sri.com, mjw@ai.sri.com

Abstract

A system is described that detects high-level events of interest in airborne motion imagery. The system starts by automatically extracting the tracks of movers from the imagery. It then uses tracks combined with context (cultural features, for example) to automatically detect low-level events. High-level events can be defined as patterns of low-level events. Exact or partial matches to these patterns of interest can be presented to an operator for further analysis.

Introduction

Airborne motion imagery is providing data at a rate that is several orders of magnitude greater than that of traditional imagery sources. Timely human analysis of this data is becoming increasingly difficult. Thus, there is a need to process this data rapidly using semi-automated analysis tools. The issue addressed in this paper is the rapid extraction and analysis of events from activity observed in airborne video (although the ideas described here apply to ground-based video as well).

Events arise through the actions of movers in the video scene. Any attempt at semi-automated analysis therefore begins by segmenting movers in the incoming video frames and organizing these observations into spatiotemporal track objects (sequences of spatial samples taken over time). Tracks can be further segmented into time points or periods that define events taken from some primitive vocabulary. For example, a mover might stop for a period of time, and this period can be marked as a STOP event. Each such primitive event can be thought of as a (possibly ephemeral) relationship among scene elements.

This paper addresses some of the challenges inherent in detecting primitive events in airborne video and assembling groups of these primitive events into more abstract composite events that are meaningful to an analyst. The process of composite event recognition should be automated in such a way as to relieve the analyst of the burden of tedious inspection, and allow more time to be devoted to the most relevant analysis tasks. The approach taken here relies in part on previously constructed event ontologies for video that define composite events using a variant on first order logic [Nev04]. Composite events are defined in terms of the relationships among scene movers, objects, zones of interest and primitive events. The extraction of composite events is compounded by the often noisy and intermittent nature of incoming data. The paper provides a look at the architecture that is used to go from pixels to composite events. The system is illustrated by the detection of an example pattern of behavior: vehicle convoying.

Architecture

The infrastructure for detecting video events consists of components for video processing, mover detection and tracking. Once tracks have been established, primitive events are extracted using site context. These events are entered into a database, where they are used by SRI's Link Analysis Workbench (LAW) to search for patterns of primitive events than match descriptions of composite events as defined by an onotology.

Figure 1: Video Light Table™, showing orthographic image (top left), video (top right), timeline (lower left) and synthetic view (lower right).

The video, tracks, and events can be browsed in SRI's Video Light Table™ (VLT), which is shown in Figure 1.

The VLT offers a synoptic view of the video and associated context. The VLT can display video, still imagery, and synthesized views, along with a timline showing tracks and events. In the timeline (lower left window in Figure 1), tracks appear as narrow green bars, while events show up as thicker multicolored rectangles. All geometry (including the UAV trajectory, the video's ground footprint, tracks and other ground features) can be selectively overlaid on the imagery. By scrolling the timeline window, the user can browse the video as well as all detected tracks and events. All objects are mouse-selectable; this allows easy retrieval of information associated with the selected object. For example, selection of an event rectangle in the timeline causes all associated tracks and objects to be highlighted in the other views.

Pixels to Tracks

We have developed and integrated a video processing system capable of detecting and tracking moving ground objects viewed from a moving airborne video camera. SRI's Moving Target Detector (MTD) system can handle general camera motions, zooms, and changes in image contrast and brightness [Hel05]. It is routinely capable of detecting moving objects with sizes down to 12 pixels (less than 2% of the linear dimension of the image). This capability can be important for detecting events such as vehicle convoys that are spread out over a large area and require a large field of view.

To detect small, low contrast moving objects, it is crucial to determine the camera motion to subpixel accuracy and the photometric changes (contrast and brightness) to within a few grey-levels. For long-standoff airborne cameras, the image motion induced by the camera can be approximated to sub-pixel accuracy by an affine transformation with six degrees of freedom. The sum of the squared pixel differences as a function of the affine parameters, using a reasonable resampling process, is sensitive to and accurately reflects very small changes in the affine parameters. Using an extension of the Lucas and Kanade method [Shi and Tomasi94] and image resolution pyramids, our target accuracy can be achieved with two or three iterations of the method at the highest resolution. Our system is further speeded up by only computing the camera motion in regions of the image with the high image texture and using integral tables for fast image inner product operations [Viola and Jones01].

The photometric changes in the camera are computed using a non-parametric technique, since the classic affine model of contrast and brightness change [Yal05] can break down near the extremes of the grey-level range. There are routinely pixels at these extremes; ground points in deep shadow and glare off of bright structures are common examples. Our method constructs a grey-level mapping between the images by computing the median mapped-to value for each grey-level and then filtering the resulting mapping function.

Ground motion is detected in a frame by comparing it to two other frames after compensating for the camera motion and photometric changes. Motion is detected at a pixel if there is sufficient change relative to both of the other frames. This ensures that the system filters out change due to disoccluded background and noise [Cut00]. Further filtering is performed using morphological operations, and the detections are grouped in space and time into trajectories of moving regions.

The moving object detection and tracking system can process 640 by 480 pixel video at eighteen frames at second on a dual processor PC. Since object motion is readily observable at much lower rates (down to 10 frames per second), our system can process video and populate a database in real time. The system has been tested on airborne video containing a total of 2,071 moving vehicles. In this test, our system achieved a detection rate for vehicles of 96% and a false alarm rate of one every two minutes. Even with these results, however, tracking remains a difficult problem due to low resolution and occlusion. Hence, a single vehicle can give rise to multiple track fragments. This underscores the importance of designing event recognition algorithms that are robust in the presence of noise and track fragmentation.

Geolocation and Site Context

The processing pipeline described in the previous section is used to construct coherent 3D tracks on the ground. SRI's FREEDIUS system is an Image Understanding system that can represent sensor models, tracks, and geospatial features in a common framework (FREEDIUS is the open-source, portable successor to similar systems developed under the DARPA IU and RADIUS programs). FREEDIUS is used for assembling MTD detections into coherent tracks, for low-level event detection, and for populating the track and event database for use by LAW. Each track is first collected and represented as a 2D curve in the image plane. Imaging geometry is obtained for each frame by using a bundle adjustment algorithm to compute camera parameters in the video. USGS or other terrain data can be used to provide a terrain model, and can be used to refine the geolocation of tracks. Once camera models have been computed, 2D tracks are projected down to the ground by intersecting the corresponding camera rays with the terrain model.

3D tracks on the ground can be processed in isolation, but this often leads to an impoverished event vocabulary for analysis. Although intrinsic track properties can be used to detect events like TURN or STOP, many events can only be detected or understood when the tracks are placed in some larger context. For example, entry into a restricted zone requires a geometric model of the zone of interest. To provide context, we use geometric site models with semantic attachment. For example, it may be desirable to create events when tracks enter a building, or traverse secure areas on the ground. This kind of

information is difficult to provide without site models and functional annotation of ground features. Non-geometric information can also be important. Entry of a person into an office building carries a different set of implications than the entry of a person into an aircraft hangar. The addition of this kind of functional knowledge to a site model provides a rich source of additional context for primitive (and composite) event detection.

Primitive Event Detection

Events are divided into two broad classes according to the ontology specified in [Nev04]. Type I events are those

in which a mover interacts with a zone or object in the environment. Events of this type include picking up or dropping off a package, entering a building, or moving inside a restricted area. Site models are used to provide the necessary context for Type I events. Type II events consist of mover-mover interactions. Two people walking side by side is one example of a type II event. One car following another is also an example of a type II event. Composite events are constructed using relationships among movers and objects in a scene that are derived from primitive events, subject to the rules provided by an ontology.

Figure 2: Convoy video example. Tracks shown in red are overlaid on the video. The window above the video shows a timeline. Small bars on the timeline show detected tracks, while large rectangles show detected FOLLOW events.

We generally define primitive events to be those events that can be extracted from a video sequence using track data and site context (including geometry). These events are detected by bottom-up processing of raw data. In contrast, composite events are detected in a top-down fashion, using patterns of interest to assemblee combinations of primitive events into composite events.

Primitive events are detected by scanning tracks, either in real-time or in a batch processing mode, and segmenting the tracks by comparing track positions with other geometric or track features in the site model. By implication, tracks themselves are considered to be spatiotemporal features entered into the site database. The track segments so generated are primitive events that are kept in the database as temporal features that define relationships among site features (either track-to-track or track-to-object).

Figure 2 shows a control panel that contains an 1800-frame video sequence of cars moving along a road. In this sequence, a stream of primitive FOLLOW events is detected using tracks obtained from the video sequence. A FOLLOW event is defined as a sequence involving two movers, separated in time, but traversing the same curve on the ground. The FOLLOW event is therefore robust in the face of arbitrary turns made by the movers.

High-Level Event Recognition

To detect higher-level events (e.g., multiple vehicles traveling in a convoy) from our database of primitive events, we are experimenting with the Link Analysis Workbench (LAW) system. LAW is a graphical pattern matcher, capable of finding inexact as well as exact matches to its patterns. A pattern in LAW's representation language, GEM (Graph Edit Model), represents two things: (1) a general description of the situation of interest, and (2) allowable deviations to that situation of interest. The situation of interest is represented in GEM as a semantic graph, and the allowable deviations are represented through parameters to LAW's similarity metric. LAW uses a graph edit distance metric to determine whether an inexact match is "close enough" to be returned to the user. The pattern author assigns costs to elements of the graph (nodes, links, and constraints)—representing the importance of finding those elements in the matching data—along with a maximum total cost—defining how inexact a match can be. The GEM language is hierarchical—each pattern may contain one or more subpatterns—and supports cardinality (requiring N or more matches to a subpattern) and disjunction. LAW's back end is a RDBMS—we have integrated it with both MySQL and Oracle—providing scalability to large data sets. LAW and its pattern language and matching approach are described more fully in [Wol03] and [Wol05].

As with many representation languages, GEM provides multiple alternative ways of representing a given situation. None of these alternatives is imperfect, and each has its own advantages and disadvantages. Here we use the problem of high-level video event detection of one particular situation—that of a convoy of vehicles—to illustrate two alternative representations and discuss their advantages and disadvantages.

Flat fixed-vehicle convoy pattern

Figure 3 shows a simple flat (non-hierarchical) pattern of a convoy involving a fixed number of vehicles, in this case five. The pattern looks for five actors related to one another through Follow events—Actor1 followed by Actor2 followed by Actor3 and so on. The Follow events are required through the "same-time" constraints to occur

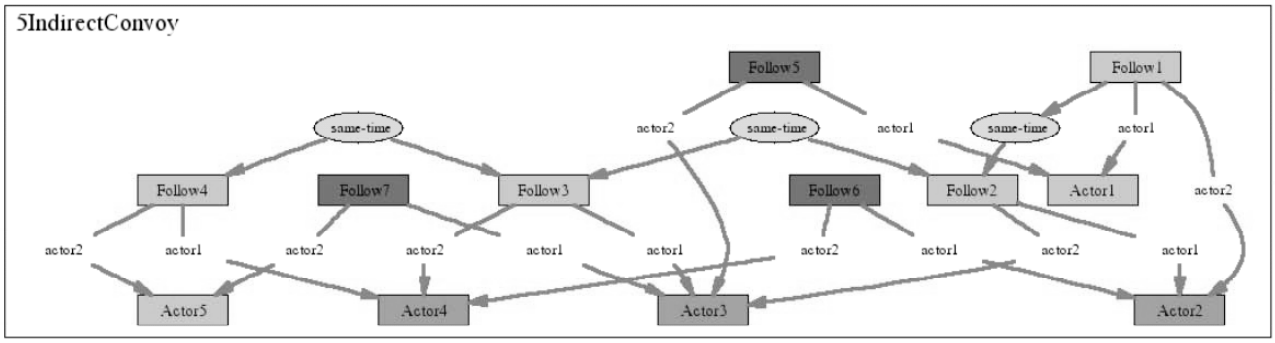


Figure 3: Flat convoy pattern with a fixed number of vehicles (5)

in overlapping time periods. In addition to the primary Follow events, the pattern also includes secondary Follow events—for example, Actor1 followed by Actor3.

This representation, combined with LAW's inexact matching criterion, supports detecting convoys in the face of incomplete data—for example, when one of the vehicles is occluded. For example, even if Actor2 becomes occluded in the video and the matcher finds no match for it, it may still be able to connect Actor1 and Actor3 through the secondary following relations. If the secondary relations were not there, there would be no contiguous matching subgraph of data to the pattern, and LAW would find no matches.

The primary advantages of this pattern stem from its simplicity. It is easy for the LAW user to understand, both the pattern and the matches from it. Additionally, it would be relatively quick pattern to author. One disadvantage of this approach to searching for convoys is that it is specific to a fixed number of vehicles; detecting any convoy of less than or (especially) more than five vehicles would require creating and matching a separate pattern.

Recursive convoy pattern

Many kinds of patterns that involve sequencing, either of events or of entities. Examples include a convoy of vehicles or a chain of phone calls (person A calls person B calls person C...). These kinds of patterns require either (1) fixing the number of participants, as does the pattern in Figure 3, or (2) some construct in the pattern language and matcher for supporting arbitrary-length iteration. LAW's mechanism for dealing with (2) is to support pattern recursion.

Figure 4 shows a recursive pattern representing a convoy of two or more vehicles. It defines a primitive convoy as either a vehicle by itself (the base case), or a vehicle following a primitive convoy (the recursive case).

This approach has the advantage of defining a convoy more realistically, in that it does not require the pattern author to specify the exact number of vehicles in the convoy ahead of time. At the same time, this pattern and matches to it will be more difficult for an end-user to understand. Further, because it is comprised of a number of small subpatterns, it is more difficult to specify to LAW how inexact matches should be treated; for this reason, this pattern will be less tolerant of occlusion and other sources of incompleteness in the data.

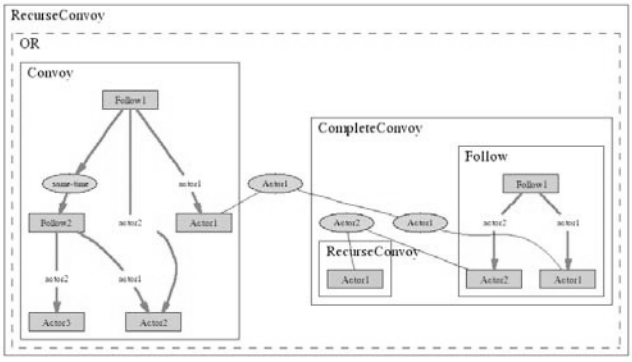
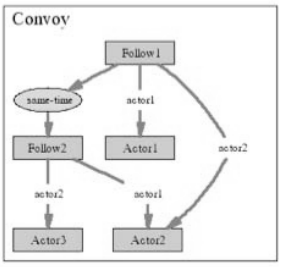


Figure 4: Graphical representation of "convoy", defined in terms of primitive "follow" events.

Experiments

We have applied the patterns in both Figure 3 and Figure 4 to three motion imagery datasets. The example in Figure 2 gives rise to two high-level convoy events. LAW returns a summary web page for these results as shown in Figure 5.



2 Matches

Convoy

Node	Value	Score
Actor1	*1153202979651256338*	1.0
Actor2	*1153202979617702007*	1.0
Actor3	*1153202979653419223*	1.0
Follow1	*1075.0 (16.0)*	1.0
Follow2	*1071.0 (6.0)*	1.0
Link		Score
1075.0 (16.0) → actor1 → *1153202979651256338*		1.0
1071.0 (6.0) → actor2 → *1153202979653419223*		1.0
1071.0 (6.0) → actor1 → *1153202979617702007*		1.0
1075.0 (16.0) → actor2 → *1153202979617702007*		1.0
Constraint	**Nodes**	Score
SAME-TIME		1.0

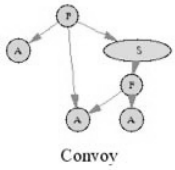


Figure 5: Two CONVOY matches for the dataset in Figure 2.

One frame of a second dataset is shown in Figure 6, and the corresponding convoy event detection is shown in Figure 7. The frame in Figure 6 is part of the interval for the convoy detection. In this example, the convoy vehicles are about to turn a corner. Figure 7 illustrates the fact that resolution can affect detection. In this case, the lead car is dropped for several frames as it decelerates around the corner. Note also that there is an extensive tree canopy in this area that can obscure the convoy at times.

In Figure 8, a third dataset is shown that illustrates convoy detection through a turn. In this case, four vehicles in the convoy are correctly detected, triggering the recursive convoy definition. One of the vehicles is obscured by tree canopy, but is detected before and after this interval. A summary of LAW detection results is shown in Figure 9.

The experiments described here represent initial steps toward an activity recognition system that is ontology-based and can flag events of interest for more detailed analysis. Many factors contribute to the correct identification of events in motion imagery. For example, the convoy pattern can be augmented to accomodate knowledge of driving patterns and cultural features. Information about ground geospatial features (such as those provided by the NGA FACC codes) can be incorporated into the system. This could allow LAW to distinguish between a genuine convoy and a line of cars at a rail crossing, for example. Knowledge of special events (e.g., parades for special occasions) could further inform LAW as to the appropriate classification of low-level events seen in video sequences.

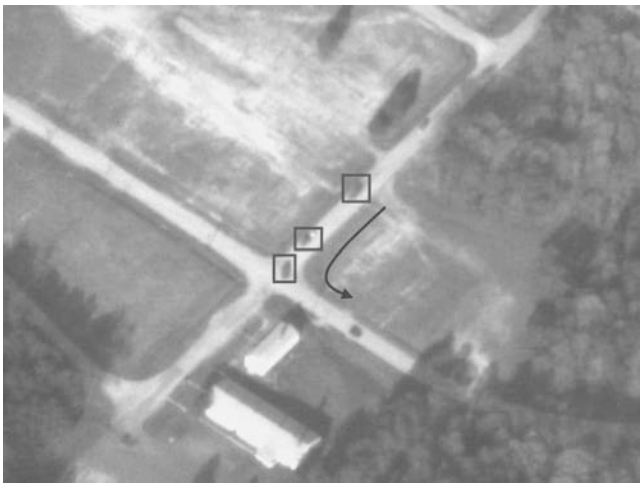
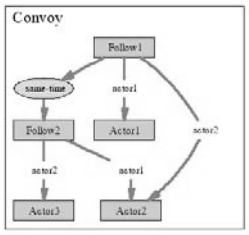


Figure 6: Convoy turning a corner. Three vehicles (highlighted) are detected as part of a CONVOY pattern.

One major benefit of using an approximate pattern matching system such as LAW is that this approach can compensate for the often noisy and fragmented quality of data coming from a visual tracker. While the tracker used in these experiments is quite robust, any tracker will fail when imaging resolution is sufficiently low, or when obscuration of the target occurs. In these cases, higher-level mechanisms can be employed to fill the gaps and to recognize and flag coherent activity in the presence of noise.



1 Matches

Convoy

Node	Value	Score
Actor1	*1153202980105617540*	1.0
Actor2	*1153202980433297421*	1.0
Actor3	*1153202980435525850*	1.0
Follow1	*12900.0 (5233.0)*	1.0
Follow2	*12967.0 (366.0)*	1.0
Link		**Score**
12900.0 (5233.0) → actor1 → *1153202980105617540*		1.0
12967.0 (366.0) → actor2 → *1153202980435525850*		1.0
12967.0 (366.0) → actor1 → *1153202980433297421*		1.0
12900.0 (5233.0) → actor2 → *1153202980433297421*		1.0
Constraint	**Nodes**	**Score**
SAME-TIME		1.0

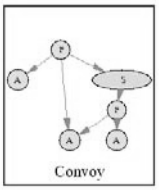


Figure 7: Match results for the dataset shown in Figure 6.

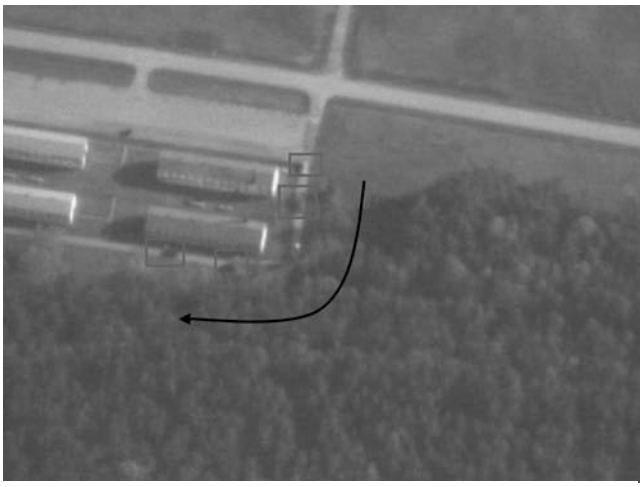


Figure 8: Convoy turning a corner in a wooded area. Four out of five vehicles are detected by the tracker. The fifth vehicle is only intermittently detected.

1 Matches

4Convoy

Node	Value	Score
Actor1	*1153202982065144026*	1.0
Actor2	*1153202982283640884*	1.0
Actor3	*1153202982301073542*	1.0
Actor4	*1153202982314180829*	1.0
Follow1	*41100.0 (1533.0)*	1.0
Follow2	*41433.0 (534.0)*	1.0
Follow3	*41633.0 (367.0)*	1.0
Link		**Score**
41100.0 (1533.0) → actor1 → *1153202982065144026*		1.0
41633.0 (367.0) → actor2 → *1153202982314180829*		1.0
41633.0 (367.0) → actor1 → *1153202982301073542*		1.0
41433.0 (534.0) → actor2 → *1153202982301073542*		1.0
41433.0 (534.0) → actor1 → *1153202982283640884*		1.0
41100.0 (1533.0) → actor2 → *1153202982283640884*		1.0
Constraint	**Nodes**	**Score**
SAME-TIME		1.0
SAME-TIME		1.0

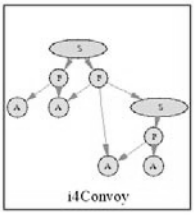


Figure 9: Details for the CONVOY match of activity shown in Figure 8.

References

[Cut00] R. Cutler and L. Davis, "Robust real-time periodic motion detection, analysis and applications", IEEE PAMI, 22(8):781-796, August, 2000.

[Hel05] A. Heller, B. Burns, et al, "Collateral Damage Avoidance and Moving Target Detection", in VIVID: Automated video processing for unmanned aircraft, T. Strat and L. Hollan, eds., DARPA, 2005.

[Nev04] J. H. R. Nevatia and B. Bolles. An ontology for video event representation. In Proc. IEEE Workshop on Event Detection and Recognition, June 2004.

[Shi94] J. Shi and C. Tomasi, "Good features to track", IEEE Conf. Computer Vision and Pattern Recognition, June 1994.

[Vio01] P. Viola, and M. Jones, "Rapid object detection using a boosted cascade of simple features, In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Dec. 2001.

[Wol03] Wolverton, M. and Berry, P. and Harrison, I. and Lowrance, J. and Morley, D. and Rodriguez, A. and Ruspini, E. and Thomere, J. “LAW: A Workbench for Approximate Pattern Matching in Relational Data.” in The Fifteenth Innovative Applications of Artificial Intelligence Conference (IAAI-03), 2003.

[Wol05] Wolverton, M. and Thomere, J. The Role of Higher-Order Constructs in the Inexact Matching of Semantic Graphs, in Proceedings of the AAAI Workshop on Link Analysis, 2005.

[Yal05] H. Yalcin, R. Collins, and M. Herbert, "Background estimation under rapid gain change in thermal imagery", in VIVID: Automated video processing for unmanned aircraft, T. Strat and L. Hollan, eds., DARPA, 2005.