Composite Spatio-Temporal Event Detection in Multi-Camera Surveillance Networks

Yun Zhai, Ying-Li Tian and Arun Hampapur

IBM Thomas J. Watson Center, Hawthorne, New York, USA

Abstract. In this paper, we present a composite event detection system for multi-camera surveillance networks. The proposed framework is able to handle correlations between primitive events that are generated from either a single camera view or multiple camera views with spatial and temporal variations. Composite events are represented in the form of full binary trees, where the leaves nodes represent primitive events, the root node represents the target composite event, and the middle nodes represent defined rules. The multi-layer design of the composite events provides a great extensibility and flexibility to users in different applications. A standardized XML-style event language is designed to describe the composite events, such that inter-agent communications and event detection module construction are conveniently achieved. In our system, a set of graphical interfaces are developed for users to easily define both primitive and high-level composite events. The proposed system is designed in the distributed form, where different components of the system can be deployed on separate processors and communicating with each other over the network. The capabilities and effectiveness of our system have been demonstrated in several real life applications.

1 Introduction

Given the rapid improving performance and decreasing costs of digital network cameras and video management systems (DVS), large scale surveillance networks along with video analytic features become more practical and affordable. Multi-camera surveillance systems require an intelligent framework for effectively extracting meaningful information, managing large-scale data, scheduling task visualization and sharing information on demand. One of the key challenges nowadays is to robustly and efficiently extract and present meaningful intelligence from multi-camera networks. Many methods have been introduced in the literature for event detection in surveillance videos from single camera views; multi-camera event detection still remains an emerging problem in real-life applications such as detection and management of large scale of complex events across multiple cameras with overlapping or disjoint fields-of-views (FOV).

In recent years, there have been many interesting work developed for detecting events or activities across multi-cameras [1][6][3][4][8][11][12]. Ahmedali and Clark [1] introduced a groundwork for a distributed network of collaborating and intelligent surveillance cameras. They proposed an unsupervised calibration technique which allows each camera module to represent its spatial relationship with other cameras in the network. For each camera, a person detector was trained using Winnow algorithm with automatically extracted training samples. To improve detection performance, multiple

cameras with overlapping FOVs collaborate to confirm results. Lim *et al*. [6] designed a scalable and wide coverage visual surveillance system by utilizing image-based information for camera control across multiple cameras. They created a scene model using a plane view of the scene. With the scene model, they can handle occlusion prediction and schedule video acquisition tasks subject to visibility constraints. Kim and Davis [4] proposed a multi-view multi-hypothesis approach for segmenting and tracking multiple people on a ground plane. To precisely locate the ground location of a person, all center vertical axes of the person across different views are mapped to the top view plane and their intersection point on the ground is estimated. Iterative segmentation-searching was incorporated into the tracking framework to deal with the explosive state space due to multiple targets and views. Remagnino *et al*. [8] proposed a multi-agent architecture for the understanding of scene dynamics merging the information streamed by multiple cameras. Wu *et al*. [11] described a framework which consists of detection, representation, and recognition for motion events based on the trajectories from multiple cameras.

More recently, composite event (i.e., events that are composed of multiple primitive events using spatiotemporal logics) detection from single or multiple cameras attracted much attention from the research community [2][5][7][9][10]. Pietzuch *et al*. [7] introduced a generic composite event detection framework that can be added on top of existing middleware architectures and provided a decomposable core language for specifying composite events. Urban *et al*. [9] addressed the development of Distributed Event Processing Agents which provide a conceptual language for the expression of composite events, an underlying XML framework for filtering and processing composite events, and a distributed execution environment for processing events from multiple streams. Bry and Eckert [2] considered a datalog-like rule language for expressing composite event queries. They have also found that the temporal relationships between events can be utilized to make the evaluation of joins more efficient. Kreibich and Sommer [5] extended the existing event model to fulfill the requirements of scalable policy-controlled distributed event management, including mechanisms for event publication, subscription, processing, propagation, and correlation. The system developed by Velipasalar *et al*. [10] allowed users to specify multiple composite events of high-complexity, and then automatically detected their occurrence based on the primary events from a single camera view or across multiple camera views. The event definitions were written to an XML file, which was then parsed and communicated to the tracking engines running on the videos of the corresponding cameras.

However, the composite event detection for multiple camera networks is usually more complicated in real world scenarios such as access violation detection, abnormal behavior analysis, large area traffic monitoring, and etc. Most of prior arts are designed for well-controlled scenes and not applicable in real-life situations. In this paper, we present a composite event detection system for multi-camera surveillance networks. The proposed system provides a generic and convenient framework for fusing primitive events that are detected by multiple sensors. The framework is designed in a scalable fashion by formulating the composite event into multi-level full binary trees. In the tree representation, the root node represent the target high-level composite event. Leaf nodes represent the primitive events that trigger the high-level composite event. The middle nodes represent the rules that combine primitive events and low-level rules with binary

operators and spatiotemporal constraints. As the essential components of a complete system that is applicable for real-life commercial use, a standardized composite event description languages is proposed to store the event definitions and transmitting events between processing agents. The description language is in the XML-style such that it is cross-platform and cross different agents. In addition, a set of graphical user interfaces are designed for providing a straightforward way to define both primitive and composite events. The proposed system is designed in the distributed form, where different components of the system can be independently deployed on separate work locations and communicate with each other over the network. We have tested our system in two real-life applications: retail loss prevention and false positive reduction. The applicabilities, effectiveness and efficiency of the proposed system are well demonstrated through the experiments.

The remainder of this paper is organized as follows: Section 2 presents the full system description of the proposed high-level composite event detection framework. Several real-life applications of our system are demonstrated in Section 3. Finally, Section 4 concludes our work.

2 Spatiotemporal Composite Event Detection

In this section, we present a high-level spatiotemporal event detection system for multi-camera networks. The composite events are formulated by combining a group of primitive events using both logical and spatiotemporal conditions. Before going into detailed description of our system, some terms need to be defined in prior. In our system, **primitive events** refer to the events that can be directly derived by analyzing video features, such as motion, color, tracking output, and etc. Some example primitive events are tripwire crossing detection, motion detection and speed detection. **Composite events**, on the other hand, refer to the events that are composed of multiple primitive events using spatiotemporal logics. For instance, a composite event "a person going entering the building" is defined as primitive event "a tripwire is crossed in camera A monitoring in front of the building" `followed by` another primitive event "a face is detected in camera B monitoring the entrance of the building lobby" within 20 seconds. Composite events are often difficult to be derived from video features directly due to limited and costly multi-camera vision technologies.

In the following sections, we present the system infrastructure of our proposed event detection framework along with event representation and detection algorithm. Then, an XML-type event description language and convenient user interfaces are introduced.

2.1 System Infrastructure

Figure 1 shows the system infrastructure of our proposed composite event detection framework. This system is designed in a way such that it is able to formulate and detect composite event targeted at either a single camera view or a combination of multiple views from a multi-camera network. There are four major components in the system. Each component is independent from others and replaceable if necessary.

– **Agent/Engine:** This is the stand alone processor for detecting primitive events in a camera view, such as trip wire, motion detection, prominent spatiotemporal curvature, etc. A single agent/engine can handle multiple primitive event detections,

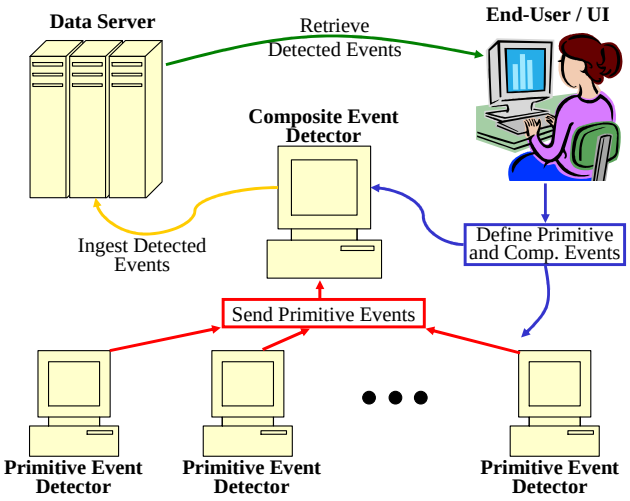


Fig. 1. Proposed high-level spatiotemporal event detection system. The primitive and composite events are defined by the end-user.

and multiple agents could reside on the same workstation. Thus, agents are able to process information that are acquired from either single camera view or multiple camera views independently.

- **Central Unit:** This is the processor for high-level composite event detections. Central unit receives detected primitive events and uses them to determine if the defined composite events occurred or not.
- **End-User Terminal:** This is the place where user defines primitive and composite events, and browses the generated results. The definition process is carried out by using a set of conveniently designed user interfaces (Section 2.4).
- **Data Server:** Once primitive and composite events are generated, they are transmitted to a centralized data server for future analysis, such as forensics of events.

A single agent can generate primitive events with multiple types, and multiple agents can reside on the same processing workstation. In addition, the *central unit* can be located on the same workstation as *agents*. The *end-user* defines primitive events (e.g., tripwire, motion, etc) and formulates target composite events using primitive events through the user interfaces (UI). The primitive and composite event definitions are then transferred to and updated on the corresponding remote *agents* and the *central unit*. Once the primitive events are detected by the agents, they will be sent over the network to the *central unit* for the composite event detection process. If a composite event is detected, summary results (e.g., alerts) are generated and stored in the *data server*, which will be available for future search and retrieval by the users.

2.2 Event Representation and Detection

The high-level composite events are represented in a full-binary tree format with one extra root node (Figure 2), where the root node represents the target composite event,

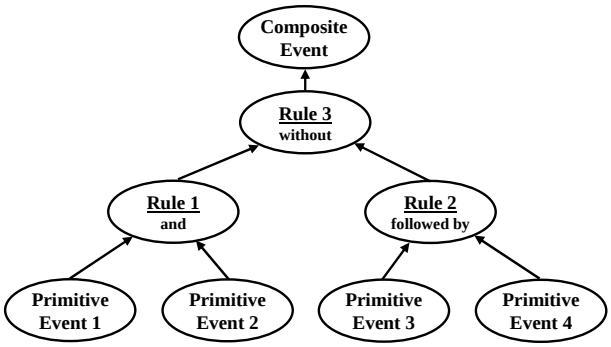


Fig. 2. Tree structure of an example composite event with four primitive events and three rules.

the leaf nodes of the tree represent the primitive events, and the intermediate nodes represent the rules. Each of the rule nodes should have a binary operator with two operands and corresponding spatiotemporal conditions. The purpose of using a tree structure is to avoid cycles, and thus, reducing implementation and computational ambiguity. If a primitive event is the operand of multiple rules, duplicate nodes for this event are created such that each rule has an independent copy. The advantage of using full binary tree form is to utilize its special property in breadth-first traversal for the composite event detection process.

In our formulation, each rule contains (but not limited to) one of the four binary operators in the following list. Each operator is coupled with a temporal condition and a spatial condition. The temporal condition is the temporal window in which the operands must be satisfied according to the operator. The spatial condition enforces the spatial distance between two detected operands, and it could either be in the image space or in the global world space. In the current system, we only focus on the spatial condition imposed in the image space. Therefore, the spatial condition is enforced only if two operands are primitive events generated from the same camera view. Each rule could be either "triggering" or "non-triggering" rule. The target composite event is detected if any of its "triggering" rules is satisfied. On the other hand, satisfaction of a "non-triggering" rule does not result in the composite event detection. It is mainly designed for building up high-level detection rules. Intuitively, every composite event should contain at least one "triggering" rule. Four proposed binary operators are:

- **AND**: This operator represents to trigger the rule if both the two operands of this operator occur within the defined temporal and spatial conditions.
- **OR**: This operator means that either one of the two operands occurs, then the rule is triggered. Due the its innate property, temporal and spatial conditions are not applied for this operator.
- **FOLLOWED BY**: This operator enforces the temporal order between two operands. The first operand must occur before the second operand to trigger the rule.
- **WITHOUT**: This operator involves the negation of the second operator. It means that the first operand should occur while the second operand DOES NOT occur within the define temporal and spatial conditions (if applicable).

There are more types of binary operators that could be incorporated. Some samples are "Event A NOT FOLLOWED BY Event B" and "Event A OR NOT Event B". They are the variations of aforementioned operators. Thus, they are omitted in our discussion.

Given above definition, a composite event can be formulated in the following form,

$$\mathbb{E} = (\mathbb{P}, \mathbb{R}), \tag{1}$$

where $\mathbb{P} = \{P_1, ..., P_n\}$ is a set of primitive events, and $\mathbb{R} = \{R_1, ..., R_m\}$ is a set of rules. Each rule R_i is defined as follows,

$$R_i = (\Phi_i^1, \Phi_i^2, \oplus_i, T_i, S_i), \tag{2}$$

where Φ_i^1 and Φ_x^2 are the two operands of the rule, $\oplus$ is the binary operator ($\oplus \in \{$`AND, OR, FOLLOWED BY, WITHOUT`$\}$), and T and S are the temporal and spatial constraints respectively. In order to enable the extensibility of the layered rule composition, the operands (Φ_x^1, Φ_x^2) could be only either primitive events or previously defined rules, i.e., $\Phi_i^* \in \{P_1, ..., P_n, R_1, ..., R_{i-1}\}$. This enables the ability of multi-layered rule composition and at the same time avoids creating cycles in the event representation tree.

Since the composite event is formulated in a full-binary tree form, it can be restructured by performing a breadth-first traversal on the tree. Eqn.1 is then reformulated as,

$$\mathbb{E} = \overline{\mathbb{S}} = \{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_{m+n}\}, \tag{3}$$

where $\overline{\mathbb{S}}$ is a vector containing the event tree nodes in the order that they are visited during the breadth-first traversal. There are two cases to initiate the composite event detection process. The first case is that the central unit receives a new generated primitive event. The second case is the idle waiting time exceeds the predefined threshold. The reason for the second initiation is that in operator "WITHOUT", negation of the second operand is used. Thus, the **non-existence** of the second operand needs to be periodically examined. Once the composite event detection process is initiated, conventional breadth-first traversal algorithm is applied to determine if any of the triggering rules is satisfied with the presence of the newly received primitive event or the non-existence of the second operand of the current rule.

2.3 Event Description Language

The composite events are stored in a standardized and extensible description language that is developed based on XML format. A composite event is represented using a two-layered description. The top layer defines the composite event, including information such as agent ID, workstation address, event name, a list of primitive events and a list of rule definitions. The lower layer is composed of two parts, primitive events and rule definitions. An example of the composite event description language is shown in Figure 3, where Figure 3.a shows the top layer composite event definition, Figure 3.b shows the definition of a primitive event, and Figure 3.c shows the definition of a rule.

In each primitive event node, tag "**<Param>**" defines the actual parameters of the event (e.g., crossing line coordinates and direction of a tripwire). In each rule definition, tag "**<Operator>**" stores the type of the binary operator, e.g., "AND" or "WITHOUT". Tag "**<Enable>**" indicates if the current rule is a triggering rule or not. Tags

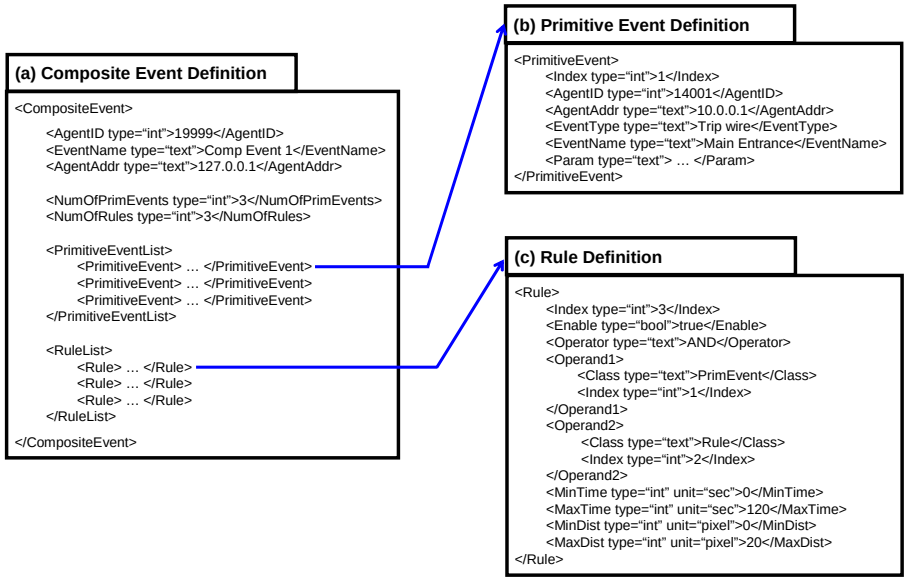


Fig. 3. Proposed high-level spatiotemporal event detection system. The primitive and composite events are defined by the end-user.

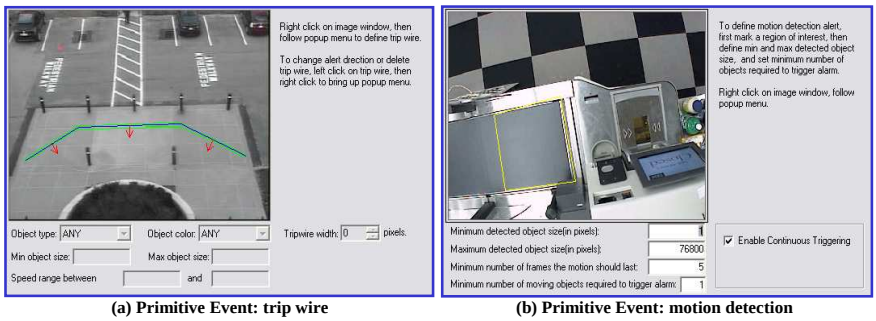
"**`<Operand1/Class>`**" and "**`<Operand2/Class>`**" indicate if the operands are primitive events or other rules. Tags "**`<Operand1/Index>`**" and "**`<Operand2/Index>`**" indicate which primitive event and/or previous rule should be the operands of the current rule. There is a tag "**`<Index>`**" in each primitive event node and rule node. This tag relates to the index tags of the rule operands. Thus, it must be a unique number to each of the primitive event and rule entities. In addition, if one of the current rule's operands is another rule, the index of the current rule must be greater than its operand's index to avoid potential cycles.

The proposed event description language is standardized, compact and non-ambiguous. This is very important in large scale camera networks, where low-traffic loads, effective inter-agent communications and efficient event constructions are required.

2.4 Primitive Events and User Interfaces

Primitive events are the building blocks of the composite events. Currently we have developed 10 primitive events, which are described below. Since our system is designed to be scalable, its capability is not limited to only these types of events. Additional primitive events and non-vision events can be easily integrated into the system (one example is shown in Section 3.1).

– **Motion Detection**: detection of motion existence within a defined region-of-interest (ROI) satisfying auxiliary conditions such as minimum motion area, motion lasting period and number of objects in the ROI.



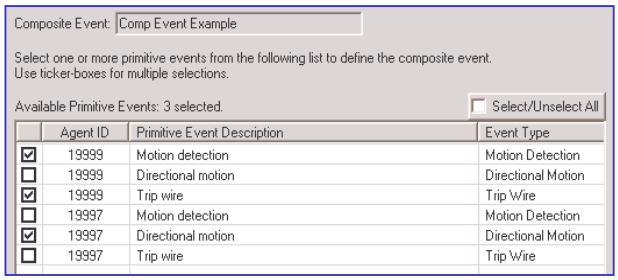
(a) Primitive Event: trip wire **(b) Primitive Event: motion detection**

Fig. 4. Screen shots of example primitive event definition editor interfaces: (a) trip wire event and (b) motion detection event.

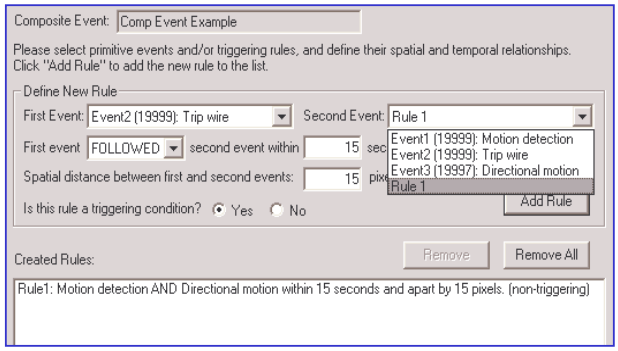
- **Directional Motion**: detection of motion presence within a defined ROI satisfying the directional filtering condition.
- **Abandoned Object**: detection of an event where an object is abandoned in the defined ROI by satisfying size and waiting time constraints.
- **Object Removal**: detection of an event where an object is removed.
- **Trip wire**: detection of an object crossing the pre-defined tripwire (composed of a series of connected line segments) satisfying both directional and size constraints.
- **Camera Moved**: detection of an event where the camera is moved to another field-of-view. This event is widely deployed on pan-tilt-zoom (PTZ) cameras.
- **Camera Move Stopped**: detection of an event where the moving camera stopped.
- **Camera Blind**: detection of an event where the camera field-of-view is lost, which could be caused by camera tampering, camera malfunction and/or network failure.
- **Face Capture**: detection of an event where faces are present in pre-defined ROIs.
- **Region Event**: detection of the directional traffic in and/or out of the defined ROI.

Primitive events are defined through a set of graphical user interfaces, where users can conveniently specify the corresponding parameters of primitive events. Screen shots of two example primitive event definition editors are shown in Figure 4. Figure 4.a shows the dialog for specifying a tripwire event, where auxiliary conditions (object size, crossing direction, moving speed, etc) are defined. In Figure 4.b, a motion detection event is specified using information such as ROIs and observed motion time conditions. Defined primitive event parameters are saved in the event description language under node "`<PrimitiveEvent/Param>`".

Once the primitive events are defined and established, users are able to formulate composite events using these primitive events through the composite event definition interfaces (Figure 5). In the first step, the system retrieves all available primitive events for the user to select (Figure 5.a). After user selects the primitive events, the composite event editor dialog will appear (Figure 5.b). The selected primitive events are the initial set of the candidates for the rule operands. To define a rule, user should select the two operands, the operator and specify the spatiotemporal constraints between the operands. Lastly, user needs to decide to enable the triggering flag for this rule or not. Once



(a) Primitive event selection dialog. Different agent IDs represent processors handling different camera views.



(b) Rule definition dialog.

Fig. 5. Composite event definition editor. User selects primitive events from dialog (a) for composite event formulation, and event rules are defined in dialog (b). Operands and operator of a new rule can be selected from the corresponding drop-down lists, and the spatiotemporal constraints are set by edit boxes. Created rules are automatically added to the operand lists.

the definition is finished, this rule is added to the rule list by clicking button "`Add Rule`". This rule is also added to the operand candidate set at the same time it is created. Intuitive descriptions of the created rules are displayed in the rule list window in a natural language style. The dynamic nature of the operand candidate set enables the capability of building multi-level event compositions. Both the primitive and composite event definition editors are implemented in the form of ActiveX components, so that they can be deployed on any workstation in the network.

3 Case Study

In this section, we present the usage of proposed composite event detection framework in two real life applications. Firstly, we propose a solution to a common problem in retail sector. Then, an example in false positive reduction is demonstrated.

3.1 Application: Retail Loss Prevention

Statistics show that in year of 2007, retailers in North America have lost over $5 billions due to various frauds. Within these frauds, a major portion is caused by *fake scans*

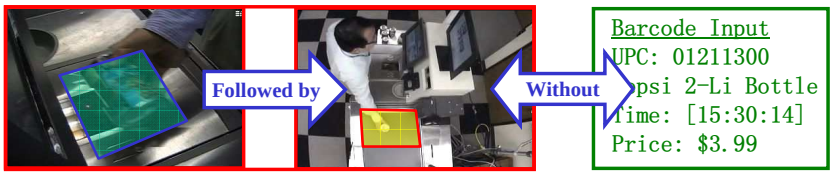


Fig. 6. Composite event formulation for detecting fake-scans in retail check-out scenarios. The composite event definition is that a fake-scan alert is triggered if, (1) a motion detection alert in barcode scanner region followed by a motion detection alert in bagging region, and (2) condition (1) happened without any barcode input. Please note that primitive events are acquired from multiple camera views, and the barcode input events are generated by the point-of-sale server which is a non-vision device.

Fig. 7. Fake-scan action detection results with different check-out lanes and different camera settings. First keyframe captures the number of fake-scans. Middle keyframe captures a particular fake-scan action. Last keyframe shows the total number of scan actions captured.

(sometimes known as *non-scans*) where the cashier (or the customer if in self-checkout) intends to by-pass the product in front of the barcode reader without actually scanning it. Here we present a solution to detect such fake scans by applying our composite event detection system on retail check-out lanes. This application demonstrates that our system is not only designed for handling visual events, but also able to integrated other types of sensor signals, such as barcode input.

The composite event is formulated as follows: two motion detection events are defined in the barcode scanner region and bagging region (belt). The two visual primitive events are detected in two different camera views to demonstrate the multi-camera capability of our system. The motion detection in barcode scanner is acquired by a camera with close-up view to the scanner, and the belt region motion detection events are acquired by an overhead camera on top of the check-out lane. These two primitive events represent object presence in designated regions. To capture a scan action, a condition is defined to compose of the motion event in barcode scanner region (denoted as P_S) followed a motion event in belt region (denoted as P_B). Temporal constraint is also imposed such that the sequence should occur within a time interval of $[1, 2]$ seconds. Once a scan action captured by the visual sensor, the system will match it with input barcode signal (denoted as P_U). If corresponding barcode signal is found within the temporal neighborhood, a fake-scan is then detected. Thus, the overall composite event for detecting fake-scans is defined as: (1) non-triggering rule R_1 := P_S `[FOLLOWED BY]`

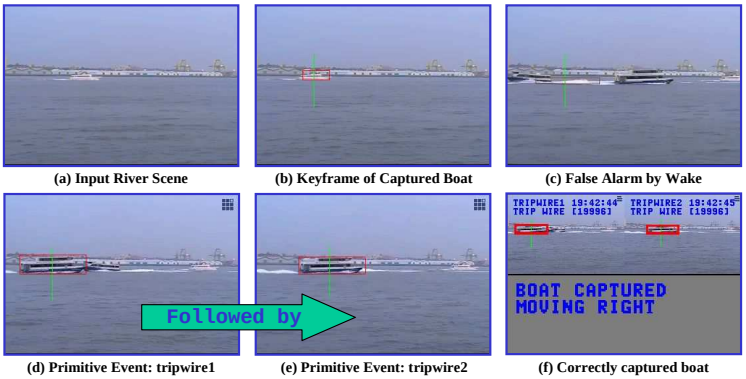


Fig. 8. (a) Keyframe of input river sequence; (b) captured boat by primitive tripwire detector; (c) false positive caused by wakes;(d) and (e) primitive tripwire detectors for the composite event modelling; and (f) keyframe of captured boat using composite event.

P_B with temporal constraint $T_1 = [1, 2]$, and (2) triggering rule R_2 := R_1 `[WITHOUT]` P_U with temporal constraint $T_2 = [0, 1]$. A graphical illustration is shown in Figure 6.

Our testing dataset consists of 9 transactions performed by different customers. Corresponding barcode signals are also available and synchronized with the video input. There are total of 179 scan actions in the dataset, where 111 are legitimate scans, and other 68 are fake scans. The system detected 96 fake scans with 5 mis-detection (false negatives) and 28 over detection (false positives). A large portion (60%) of the false positives are double-triggers. This accuracy has greatly exceeded industry expectations. ("Dropping shrinkage from 2% to 1% of sales has the same effect on profit as 40% increase in sales", AMR 2004). Figure 7 shows some keyframes of captured fake scans.

3.2 Application: False Positive Reduction

As mentioned in previous application, primitive event detector sometimes generate false positives due to limitations in technology and/or complex physical settings of the scene. In this section, we discuss the application of using composite event to reduce false positives generated by primitive event detector.

Consider the application of monitoring traffic on water body (Figure 8.a) where statistics of boats travelling in certain direction needs to be recorded daily. In this case, tripwires can be defined on the water surface to capture boats that are moving in the target direction (Figure 8.b). As commonly known, reliable object detection and tracking on water is a difficult problem due to the uncertain pattern of water waves and/or wakes caused by boats. In this case, false positives are likely to occur for single tripwires (Figure 8.c). To resolve this problem, and thus reduce the false positives, we utilized the fact that boat wake often only appears for a very short period or time. Unlike boats, wake does not travel consistently on the water surface for a long distance. Therefore, if two tripwires are defined in consecutive order, they are able to capture the boats since only boats can cross both the tripwires and wakes can not. Thus, a composite event is formulated as a tripwire P_X followed by another tripwire P_Y within certain time period. Due

to the nature of this problem, both tripwires are defined within the same camera view. They have the same crossing direction defined to ensure that they should be triggered by the same boat. The composite event definition for capturing boats on river is illustrated in Figure 8. One result keyframe of captured boat is shown in Figure 8.f without the false positive shown in Figure 8.c.

4 Conclusions and Future Improvements

In this paper, we have presented a high-level spatiotemporal event detection system for multi-camera networks. The system integrates primitive event detectors into composite events. The composite events are formulated as full-binary trees where leaf nodes are the primitive event detectors and middle nodes represent the rules. A standardized description language is proposed for transmitting and storing the composite event definitions. The applicability and effectiveness of the proposed event detection system have been demonstrated in a variety of real-life applications. Current framework models the rules in composite event as binary operators (e.g., AND, OR). In future work, we plan to extend our formulation such that unary and multi-operand operators can be also integrated. Another future improvement will be incorporating high-level auxiliary constraints to the rules. For instance, primitive events can be combined together if they are triggered by objects with the same identity.

References

1. T. Ahmedali and J.J. Clark, "Collaborative multi-camera surveillance with automated person detection", *Canadian Conference on Computer and Robot Vision*, 2006.
2. F. Bry and M. Eckert, "Temporal order optimizations of incremental joins for composite event detection", *Proceedings of the 2007 conference on Distributed Event-based Systems*, 2007.
3. V. Kettnaker and R. Zabih, "Bayesian Multi-camera Surveillance", *CVPR*, 1999.
4. K. Kim and L.S. Davis, "Multi-camera Tracking and Segmentation of Occluded People on Ground Plane Using Search-Guided Particle Filtering", *ECCV*, 2006.
5. C. Kreibich and R. Sommer, "Policy-controlled event management for distributed intrusion detection", *Workshop of Conference on Distributed Computing Systems*, 2005.
6. S. Lim, L.S. Davis and A. Elgammal, "A Scalable Image-Based Multi-Camera Visual Surveillance System", *Conf. on Advanced Video and Signal Based Surveillance*, 2003.
7. P.R. Pietzuch, B.Shand and J. Bacon, "Composite event detection as a generic middleware extension", *IEEE Network*, 2004.
8. P. Remagnino, A.I. Shihab and G.A. Jones, "Distributed intelligence for multi-camera visual surveillance", *Pattern recognition*, 2004.
9. S.D. Urban, S.W. Dietrich and Y. Chen, "An XML Framework for Integrating Continuous Queries, Composite Event Detection, and Database Condition Monitoring for Multiple Data Streams", *Event Processing*, 2007.
10. S. Velipasalar, L. Brown and A. Hampapur, "Specifying, Interpreting and Detecting High-level, Spatio-Temporal Composite Events in Single and Multi-Camera Systems", *Int'l Workshop on Semantic Learning Applications in Multimedia (SLAM), in the Proceedings of IEEE CVPR Workshop, New York*, 2006.
11. G. Wu, Y. Wu, L. Jiao, Y. Wang and E.Y. Chang, "Multi-camera Spatio-temporal Fusion and Biased Sequence-data Learning for Security Surveillance", *Proceedings of the eleventh ACM international conference on Multimedia*, 2003.
12. H. Zhou and D. Kimber, "Unusual Event Detection via Multi-camera Video Mining", *Int'l Workshop on ICPR*, 2006.