PROVIDING PHYSICAL SECURITY VIA VIDEO EVENT AWARENESS

Dimitrios Georgakopoulos and Donald Baker
Telcordia Technologies, Austin Research Center, 106 E. Sixth Street, Suite 415, Austin, Texas 78701, USA

Keywords: Video surveillance, event processing, situation awareness.

Abstract: The Video Event Awareness System (VEAS) analyzes surveillance video from thousands of video cameras and automatically detects complex events in near real-time—at pace with their input video streams. For events of interest to security personnel, VEAS generates and routes alerts and related video evidence to subscribing security personnel that facilitate decision making and timely response. In this paper we introduce VEAS's novel publish/subscribe run-time system architecture and describe VEAS's event detection approach. Event processing in VEAS is driven by user-authored awareness specifications that define patterns of inter-connected spatio-temporal event stream operators that consume and produce facility-specific events described in VEAS's surveillance ontology. We describe how VEAS integrates and orchestrates continuous and tasked video analysis algorithms (e.g., for entity tracking and identification), how it fuses events from multiple sources and algorithms in an installation-specific entity model, how it can proactively seek additional information by tasking video analysis algorithms and security personnel to provide it, and how it deals with late arriving information due to out-of-band video analysis tasks and overhead. We use examples from the physical security domain, and discuss related and future work.

1 INTRODUCTION

Heightened security demand for physical security has led to the deployment of large numbers of surveillance cameras. However, the resulting large volumes of video will not improve security without the development of more advanced surveillance technology and new practices. Although some surveillance systems currently employ limited video analysis automation to facilitate detection of simple video events (i.e., events in the view of a single camera, such as detected motion in a room or a security perimeter), virtually all existing surveillance systems rely mainly on human operators for the detection of real-world situations of interest that are *complex events* involving patterns of simple video events distributed in time and possibly occurring in different locations in a facility. Therefore, forensic event analysis performed by human operators is currently the main (often unstated) reason for deploying security cameras.

This work was supported in part by the Advanced Research and Development Activity (ARDA), Video Analysis and Content Extraction (VACE) Program under contract HM1582-04-C-0007.

One possible solution for taking advantage of the large volume of raw surveillance video is to increase dramatically the number of human operators monitoring a facility. However, this solution is typically beyond existing organizational resources and budgets. Furthermore, adding cameras and operators may reduce detection reliability. This is because it is difficult for humans to reliably relate events that occur at different times and in different spaces in a facility, as they are typically shown in different video displays (as illustrated in Figure 1) and at different times.

Figure 1: Typical bank of monitors that security personnel might use for monitoring a facility.

Georgakopoulos D. and Baker D. (2006).
PROVIDING PHYSICAL SECURITY VIA VIDEO EVENT AWARENESS.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 240-251
DOI: 10.5220/0001573202400251

The goal of the Video Event Awareness System (VEAS) is to analyze surveillance videos from thousand of cameras and other non-video sensors and *automatically* detect complex events that indicate situations of interest, *alert* humans about them, provide the *evidence* that led to the alerts, and do this in *near real-time*—at pace with their input video streams.

To meet these goals, VEAS has been designed to solve the following engineering problems.

- Correlating information from multiple video and non-video sources into a coherent picture of the activities of people within a facility.
- Providing near real-time event detection from continuously run streaming video analysis algorithms.
- Proactively seeking additional information when there is uncertainty or gaps, usually by tasking video analysis algorithms, and incorporating this late arriving information into updated alerts.
- Enabling resource optimization so that expensive video analysis algorithms are run only when they are likely to yield useful information.
- Facilitating situation understanding so that security personnel can grasp the circumstances surrounding an alert and react more appropriately.
- Permitting situation-specific and installation-specific customization so that the system can be effectively used in multiple installations whose needs may change over time.

In this paper, we focus on VEAS's run-time architecture and describe its event processing model and engines. Although we only show how these innovations enable the automation of surveillance, the proposed architectures and engines are general enough to automate event detection and provide situation awareness in any sensor-based system. The rest of this paper is organized as follows: Section 2 provides an overview of the VEAS architecture and introduces a running example. Sections 3-6 discuss the main components of the VEAS architecture and explain how each component helps to address the engineering problems we described earlier in this section. Related work is presented in Section 7, and conclusions in Section 8.

2 VEAS OVERVIEW

In the following sections we describe the main components in the VEAS architecture and present an example of a security policy that can be specified and automated by VEAS.

2.1 VEAS Architecture

The main contribution of this paper is the introduction of the novel VEAS architecture that meets the goals of automated near real-time facility surveillance. As depicted in Figure 2, the architecture is comprised of a sequence of information processing components that together distill vast amounts of data from video and other non-video sensors (e.g., RFID and badge readers) into useful information in the form of alerts that are of interest to security personnel. The arrows in Figure 2 indicate the primary information flow between the VEAS components. In addition to information flow, the arrows indicate (right to left)

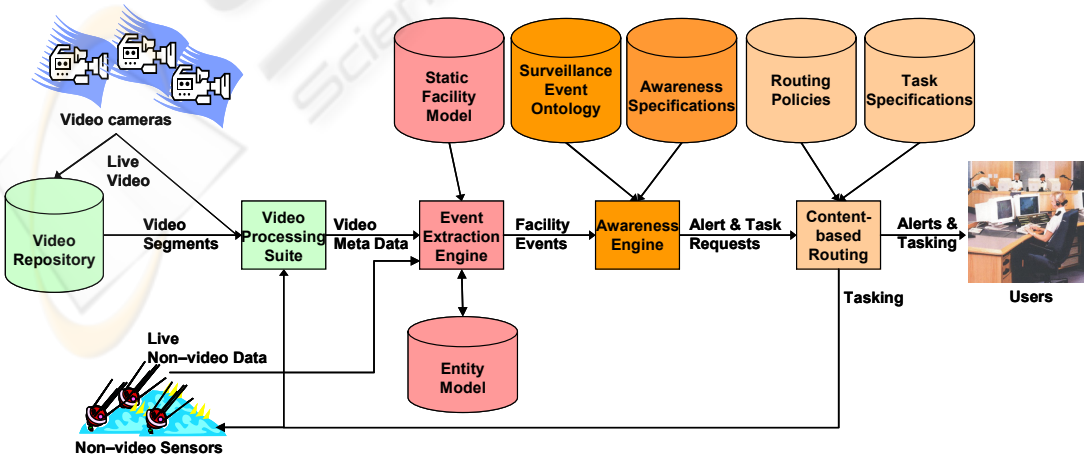

Figure 2: The VEAS run-time architecture.

client-server relationships and data subscription relationships between components. The data made available to each component triggers its computation, producing data for the next component in the sequence.

Live video flows from the video sources to both the *Video Repository* (VR) component and the *Video Processing Suite* (VP Suite) for real-time analysis by streaming video analysis algorithms. The VR segments video streams into manageable file sizes using a time-based video segmentation approach (e.g., as in (Moser, et al., 1995; Aref, et al., 2002)), and stores them in the file system. The VP suite provides video analysis algorithms (e.g., for entity tracking, face detection) that produce their results (e.g., entity tracks, video frames with faces) in *near real-time*—at pace with their input video stream. The VP Suite can additionally extract information from video segments stored in the VR upon request from its client components. This approach is used for expensive video analysis algorithms (e.g., face recognition) that operate on stored video segments and may not produce their results in near real-time. The delay caused by tasking video analysis algorithms causes of information to be processed (possibly long) after the time of the real-world situation was captured on video. We call such information *late arriving*. Since information from non-video sensors require no processing, the VEAS architecture lacks a processing suite for such data.

The purpose of the *Event Extraction Engine* (EEE) is to collect video metadata and other data from non-video sensors, and use these to construct an *Entity Model* (EM). The EM is initialized and subsequently updated by the *Static Facility Model* (SFM) as depicted in Figure 2. The SFM is maintained by VEAS administrators. It captures site-specific information that includes:

- Spatial descriptions of the *space* hierarchy in the facility and security related attributes for each space, such as its intended purpose (e.g., office, meeting room, record vault) and access restrictions (e.g., escorted visitors, employees with a specific clearance).
- Information about known entities (e.g., employees, contractors) and their security related attributes (e.g., name, photo, office, security clearance).

The Entity Model is an extensible model of the facility, its occupants, and their activities (e.g. movements, behaviors, etc.) that is continuously updated as new information becomes available. To populate the EM, VEAS's EEE utilizes the streams of video data from the VP Suite and non-video sources (e.g., RFID tags and badge readers). Additional changes to the EM occur, for instance, when humans enter, are identified, move around, and leave the facility. Each of these changes constitutes a *facility event*. If a facility event is of interest to the Awareness Engine, it is forwarded to the AE for consumption.

The *Awareness Engine* (AE) performs continuous detection of complex events as specified by event patterns called *awareness specifications*, and generates an alert when an event pattern in an awareness specification is matched. Awareness specifications are authored in concert with the *Surveillance Event Ontology* that provides a semantic type system for events of interest. Awareness specifications and the Surveillance Event Ontology are stored in their respective repositories as illustrated in Figure 2 and they are described further in Section 4.

The AE and EEE work together to process awareness specifications as follows: First, AE performs a decomposition of event patterns in awareness specifications to determine their constituent facility events, and issues subscriptions to these facility events from the EEE. The EEE decomposes the subscribed facility events to desired video metadata over (video source, video analysis algorithm) pairs, and then performs continuous analysis of video streams and extraction of the metadata. The AE consumes facility events to perform continuous detection of event patterns that has been specified in awareness specifications. Finally, the AE generates alert requests when the patterns are matched, which flow to the Content-based Routing Engine. In addition to generating alerts requests, the AE generates task requests for proactive information gathering tasks, which also flow to the Content-based Routing Engine. Task requests are generated only in response to specific situations as described in an awareness specification. This specificity effectively limits the cases in which an expensive video analysis task may be run, thus optimizing computing resources.

The *Content-based Routing Engine* (CBRE) interacts with the Awareness Engine to route alerts to appropriate security personnel. Delivered alerts come with their pedigree and supporting video evidence so that users can understand the cause and greater context of the alert. This facilitates situation understanding. The CBRE also tasks video processing algorithms that gather information whenever an event pattern indicates a situation requiring further information as indicated by the AE. *Task Specifications* are defined by VEAS

administrators and stored in the repository shown in Figure 2.

The following section presents a security policy example we use throughout this paper to describe VEAS's functionality.

2.2 Example Security Policy

Consider the detection of unescorted visitors within any space in a facility that requires that visitors be escorted. For the sake of this example, we'll divide people into the following disjoint categories:

- *employees*, who must either badge-in when they enter the facility or they must sign-in with the receptionist to verify their identity and receive a temporary badge; and
- *visitors*, who must sign in at the receptionist desk before they enter the rest of the facility.

Visitors have assigned escorts, who might often be the employees they are visiting. The receptionist checks the visitor identity and consults a log of expected visitors and employees to make the escort assignment for the visitor.

Our example policy considers a person as *escorted* if he/she is in the same space (i.e. room) as one of their assigned escorts. Not all rooms in our facility require visitors to be escorted. In particular, the facility model designates each space in the facility as either prohibiting visitors, allowing visitors unescorted, or requiring visitor escort.

Our example security policy defines a visitor to be unescorted if (1) the visitor is in a room that requires visitor escorts, and (2) the visitor remains in this room without any designated escort for longer than a period of time, say 30 seconds. Introduction of a time bound tolerance reduces false alarms because a visitor and his escort may not transition between rooms at the same time as they move about.

While employees serving as escort know that they are responsible for escorting their visitor, they may not always follow the security policy. In a facility with more than a few visitors, security personnel may have great difficulty reliably detecting this security violation as a visitor may be considered unescorted even if they are in a room with people who happen not to be designated escorts. VEAS can reliably detect this real-world situation and alert security personnel appropriately.

In the following major sections, we will describe the operation of VEAS's components and we will use the unescorted visitor security policy to discuss each component's contribution to the detection of security policy violations.

3 VIDEO PROCESSING SUITE

The Video Processing (VP) Suite is actually a collection of separate components, all of which extract *video metadata* from video streams or stored clips. Each VP component performs two functions:

- it integrates an existing video analysis algorithm so that that acts as a consumer of live video streams or video segments from the Video Repository; and
- it publishes its results to the Event Extraction Engine as it becomes available.

Algorithms in the VP Suite can be divided into the following two broad categories:

- *Continuous* video analysis algorithms can be run as video stream filters that perform video processing in a continuous fashion. Continuous video analysis algorithms are directly connected to the live video sources.
- *Tasked* video analysis algorithms run only in specific circumstances and they usually take video segments as input.

Regardless of whether the VP algorithms are continuous or tasked, they all publish their results to the Event Extraction Engine as they are produced.

For VEAS to work properly, the VP Suite should contain at least one continuous video analysis algorithm for *tracking entities* (i.e., humans and other movable objects). The number and location of the cameras whose video streams are processes by this continuous algorithm should be sufficient to enable VEAS to gather basic knowledge of what *entities* are in the facility and how they move about.

For the unescorted visitor example, VEAS uses continuous tracking to follow entities within each camera's view. The VP publishes these single camera tracks as they are collected to the Event Extraction Engine (EEE). The EEE continuously stitches these tracks together and correlates them with identity information gathered from other sources. VEAS can determine the whereabouts of any visitor and his/her escorts in near real-time.

Tasked VP Suite algorithms are assumed more expensive than their continuous algorithm counterparts. Typically, tasked video analysis algorithms are run only in situations where the information potentially gathered by the algorithm justifies the computational expense of running it. Therefore, effective tasking a tasked video analysis requires any clients of the VP suite to first identify the situation where the algorithm might be useful and then perform the actual invocation of the algorithm for that particular situation. The VP Suite handles the invocation and results publication for the

video analysis algorithm using a web services platform.

The collection of video analysis algorithms actually used in the VP Suite may depend on the security policies of each VEAS installation, the availability of algorithms and computational resources needed to run them, and the expected costs of their invocations. We are currently utilizing algorithms from the U.S. Government's DTO (formerly ARDA) VACE program (VACE 2006).

4 EVENT EXTRACTION ENGINE

The purpose of the Event Extraction Engine (EEE) is to perform continuous analysis of video streams (as well as data streams from non-video information sources) and extract simple events about the facility being monitored, called *facility events*. To do this, the EEE maintains as its primary information asset an *Entity Model* (EM). The EM is a continuously extended model of the facility, its occupants, and their activities.

The EM is a key VEAS innovation. In particular, EM permits simple, high-level awareness specifications, since these are built on top of the EM which acts as a "world" model of the facility. In addition, EM provides a simple, yet effective, strategy for video analysis efficiency, since it avoids unnecessary video analysis by combining and caching all known information for a facility.

The specific information kept in the EM is described in Section 4.1. The construction of EM is described in Section 4.2. The use of EM by the Awareness Engine is discussed in Section 5.

4.1 Information in the Entity Model

The Entity Model maintains a dynamic record of *entities* (i.e., humans and other movable objects) and their movements in space and time. In particular, the EM contains the following tracking and identification information: *spaces* (e.g., rooms, portals, etc.), *entities, real world identities of people, current and historical relationships between these, and time intervals when these relationships occur.*

The EM fuses information from various video analysis algorithms. More importantly, EM continuously builds three dimensions of facility information whose purpose is to allow the Awareness Engine (AE) to determine instantaneously the following:

- the location of everyone in the facility at a particular time (e.g., 1:22pm);
- the pattern of use in a particular room (e.g., Room 4); and
- how a particular person (e.g., John Smith) moves around the facility over time.

Based on the availability of more advanced video analysis algorithms in the VP Suite the EM can be extended easily to include additional details to make a broader or more detailed model of the facility. Possible extensions include the following:

- Extending the spaces to larger or more complex sites and areas, such as battle zones, cities, etc.
- Providing state information related to individuals under surveillance (Hongeng, et al., 2003), e.g., body position, gestures, gaze, etc.
- Tracking inanimate entities, such as briefcases, packages, vehicles, shipping containers, etc.
- Tracking entity-entity interactions, such as conversations, hand-shakes, carried bags, vehicle collisions, etc.
- Incorporating additional sensor information, such as temperature, sound, chemical or radiation detection, heart rate, etc.
- Modeling normalcy such as patterns of individuals' behavior, room usage statistics, social network analysis, etc.

These EM extensions are beyond the scope of this paper, and they are not discussed further.

4.2 Entity Model Management

To construct the EM, the EEE does the following:

- pulls information from the Static Facility Model (SFM) that has been populated by surveillance administrators;
- utilizes the video metadata (produced by both streaming and tasked video analysis algorithms in the VP Suite);
- utilizes the streams of information produced by non-video sources; and
- integrates these in an incremental fashion to make a coherent picture of what is currently known about the facility and its occupants.

The SFM (illustrated in the VEAS architecture in Figure 2) is the source of many installation-specific, space- and people-related facts in the EM. In particular, the SFM captures information about the facility, its hierarchy of spaces, boundaries between those spaces, camera locations and orientations, and identities of known people. Additional SFM facts include the type of each space (e.g., office, conference room, lab, hall way), the association of office spaces to individuals and shared

spaces to organizations, and the spaces that are off-limits to each class of entities. The SFM also provides facts about known persons. Examples of information about a person include his/her name, employee ID (if he/she is an employee), photos for face recognition, etc.

Non-video sensors (e.g. RFID readers, badge scanners) typically provide information as to the identity of moving entities and their position in the facility as they move about in time. Such information streams can be correlated in the EM with similar information collected from video.

EEE fuses all its input information in facility-wide entity tracks, and inferences cross-track entity IDs even in situations where entities move through areas that have no camera coverage. EEE also incorporates the real-world identities of the entities.

To perform facility-wide entity tracking, the EEE keeps video source independent information in the EM concerning how entities move through the facility. This information is camera independent and makes no assumptions about the real-world identity of persons. Tracking algorithms generate entity IDs when they detect a moving entity. Facility-wide tracking can be performed at different, predefined levels of spatial granularity. Supported granularities in VEAS include buildings, rooms (both captured in the static facility model), and global (grid-based) coordinates. For a given entity ID and interval of time, the EEE records in the EM a sequence of tuples indicating that the person spent a particular interval of time in a particular space. The entire sequence also has a pedigree summarizing the sources considered in generating the sequence and a certainty measure. Each sequence is contiguous and has a unique entity ID as long as the Event Extraction Engine has a high confidence of the continuity of identity.

Cross-track entity ID inference is necessary when blind spots exist in the facility, e.g., rooms without cameras. In this case, there may be circumstances where it is not known with certainty whether the entity ID associated with one track is the same as that of another track. To infer the real-world identity of entities the EEE may use one of the following solutions:

- Correlate a person seen in a video segment with the time and place of a badge scan or RFID reading at that location.
- Task face recognition or other identification algorithms in the VP Suite to determine the identity of persons appearing in a video segment through comparison to a set of known faces (such information for employees may be included in the Static Facility Model).
- Utilize assessments made by security personnel tasked with determining the identity of a person through viewing a video segment or via direct interaction with the person in question.

In the last two solutions, tasking is initiated by the Awareness Engine, as described in Section 4.

For the unescorted visitor example, the EEE tracks the locations and identities of all people as they move around the facility. The EM utilizes information from the Static Facility Model to determine how various spaces are designated as to whether visitor escort is required within them, and the real world identities of known entities, such as the employees. As people enter the facility, badge scans and receptionist records published into the EEE are correlated with the video tracks of entities as they move around the facility. The EM is also capable of recording the designated escorts for each visitor. This information is provided to the EEE via the receptionist.

The Event Extraction Engine publishes the EM incrementally, i.e., as video and non-video information is streamed to the EEE and is published within in the EM. The Awareness Engine can register interest in any particular combination of persons, spaces, time intervals and receives updates in the form of facility events.

The EM is maintained as a 3-tiered storage solution. In particular, the EM is kept in a main memory data structure that serves as a cache for an EM database that is also periodically archived. Transition of information between the EM storage tiers is based on installation-specific policies that consider the age of the information and whether it contributes to alerts that are still of interest to security personnel.

5 AWARENESS ENGINE

The Awareness Engine (AE), illustrated in Figure 2, consumes facility events published by the Event Extraction Engine and detects complex events. When it detects such events, the AE publishes *alert requests* and *task requests* that flow to the Content-based Routing Engine for execution. To detect complex event and compute corresponding alert and task requests, the AE utilizes user-authored *awareness specifications* that describe how to compute complex events of interest and who should receive an alert or perform a task.

In Section 5.1, we introduce VEAS's event ontology and awareness specifications. Section 5.2 describes the types of event stream operators that exist in VEAS and discusses how event operator types can be created and extended. Section 5.3 describes the alert and tasking event operators. In these sections, we provide a detailed example of an awareness specification for the unescorted visitor security policy.

5.1 Awareness Specifications

Awareness specifications are comprised of customized, interconnected computational units called *event operators*. Figure 3 shows such a specification for detecting unescorted visitors.

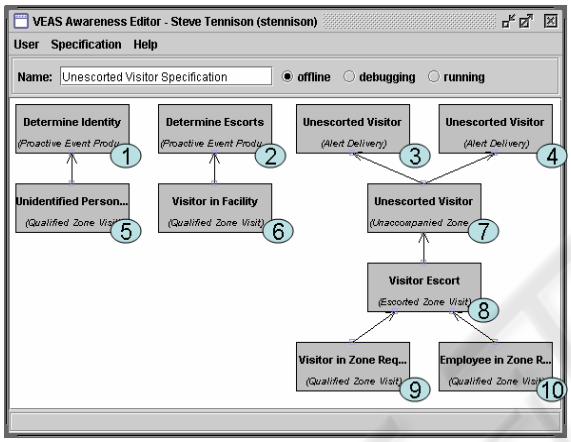

Figure 3: Example awareness specification utilizing operators 1-10.

Event operators in Figure 3 appear as boxes with a descriptive name. Ovals in the figure number the event operators for reference in this paper. Event operators are connected into computational pipelines that operate in parallel. Within each pipeline, events flow from the bottom to the top. The inputs and outputs of event operators are typed streams of *events* (i.e., packets of information indicating real-world situations).

Event types of interest, their relationships, and the information computed for each event type is described in the *Surveillance Event Ontology* (not shown in Figure 3). VEAS's Surveillance Event Ontology (SEO) acts as a type system describing the events of interest for a specific installation of VEAS. For example, the *Zone Visit* event type described in the SEO includes information about the space within the facility being visited; the entity ID of the visitor; the real-world identity of the visitor, should the visitor be a person; and the time interval of the visit.

Event operators are computational units that recognize more complex real-world events (produced on their output stream) from simpler real-world events (consumed on their input stream(s)). A connection between the output of one operator and the input of another is only allowed if they have semantically compatible event types, as described by the Surveillance Event Ontology. This restriction helps ensure information compatibility between producing and consuming event operators, but it also ensures semantic compatibility, so that the overall awareness specification computes alert and task requests relative to the author's understanding of the real-world event being specified.

To meet the surveillance needs of a particular installation (e.g., to support the specific security policies that need to be enforced), VEAS provides a tool called the *Awareness Specification Editor* that permits authorized VEAS users to:

- Customize the Surveillance Event Ontology by creating new/refining existing event types.
- Author awareness specifications by inter-connecting and customizing event operators via dialog boxes.

The awareness specification in Figure 3 is shown in this tool.

To aid in the development of new operators that cannot be produced by customizing existing ones, VEAS provides a programming interface. With proper training, new event operator types can be developed by a programmer in a matter of hours.

VEAS permits the creation of installation-specific operator palettes that are aimed to simplify authoring of awareness specification in each specific VEAS deployment. Such pallets typically include custom operator types with names closely resembling the main concepts in the security policies of a specific installation. In Sections 5.2 and 5.3, we discuss various operators and give examples.

5.2 Event Stream Operators

VEAS provides two generic *spatio-temporal* operator types that use the Entity Model as their tacit input. These operators (and customized operators based on them) commonly appear as event stream sources within virtually all awareness specifications. The *Qualified Zone Visit* operator type allows the specification of simple events qualified by time, space, identity, or attributes thereof for instances of people known to be in spaces throughout the facility and produces events of the Zone Visit event type, described earlier. A similar event operator type,

Qualified Portal Traversal, recognizes people traversing specific portals within the facility.

There are four *Qualified Zone Visit* event operators appearing in the unescorted visitor awareness specification in Figure 3. These operators have been customized to recognize (from top to bottom and left to right):

- situations where an unidentified person is in the lobby of the facility where the receptionist sits (operator 5);
- situations where a visitor is known to be somewhere in the facility (operator 6);
- situations where a visitor is known to be in a zone where visitor escort is required (operator 9); and
- situations where an employee is known to be in a zone where visitor escort is required (operator 10).

The title of the event operator is displayed and each operator is customized via dialog boxes. Events meeting a specification flow (upward in the specification) via event operator interconnections.

VEAS provides a few *set* operator types that perform set functions over their input operator streams. The *Or* event operator type computes a union of its input event streams. The *Difference* event operator type computes a set difference of two input streams. These operators are polymorphic with respect to their input and output event types.

The remaining event stream operator types can be categorized in broad functional categories:

- *joining* – combining related events from multiple input streams into composite events on the output stream;
- *filtering* – culling of uninteresting events from the input in the output; and
- *grouping and aggregation* – grouping of multiple input events from a single stream into aggregated output events.

Join event operators behave like a join operator in a stream query in that they merge information from multiple sources. The *Accompanied Zone Visit* operator type, for example, joins two input streams of Zone Visit events. The first input stream describes people in zones for which accompaniment information is desired. The second input stream describes people in zones who are possible accompanists. For each event on its first input, the operator outputs an event that adds information about whether the person was accompanied, by whom, during what time intervals, and the intervals during which he or she was unaccompanied.

For the unescorted visitor example, we created a variant of Accompanied Zone Visit operator type. The *Escorted Zone Visit* (operator 8) is an installation-specific join operator customized to find time intervals where both a visitor and one or more of his escorts are in the same zone at the same time. The operator computes the associated escorts, the escorted time intervals, and the unescorted time intervals, and the longest unescorted time interval as part of its output. These values are computed regardless of whether any escorts are found for the visitor. Visitor zone visits to be considered are provided on the first input from event operator 9. Possible escort zone visits to consider are input as the second input from event operator 10.

Filtering operator types are the simplest category in that they output events from their input event stream that meet the filter criteria as specified by the event operator's customization. In the unescorted visitor example, the *Unescorted Visitor* (operator 7) is an installation-specific filter event operator that looks for escorted zone visits on the input with a longest unescorted interval that is greater than the specified threshold. In this specification, the threshold is set to 30 seconds via a customization dialog box.

An example of an *aggregation* is the *Fleeing* operator type (not used in the awareness specification in Figure 3). The operator produces output events that represent episodes of people exiting a particular space meeting the criteria of a minimum count of people exiting and a minimum sustained exit rate over the episode. The operator is used to look for mass exodus events in a facility that may or may not be accompanied by alarms or other signals known to security personnel. The Fleeing event operator type is an example of an event operator that embodies a computation that is difficult or impossible to express in a stream query language. Its notion of grouping is beyond the capability of standard "group by" constructs as the group is not readily indexed. It would be impossible to capture this notion of fleeing in a simple streaming query.

5.3 Alert and Tasking Operators

The *Alert Delivery* and *Information Gathering* operator types submit prioritized alert and information gathering task requests, respectively, to the Content-based Routing Engine. Operators of these types appear as event stream sinks in awareness specifications. Both alert requests and tasks requests have a priority value that is incrementally computed in these two operators by a threat analysis module based on the input event type and event instance information.

The *Alert Delivery* operator converts its input event stream into prioritized alert requests to be sent to the Content-based Routing Engine. User customization of this operator effectively provides delivery instructions that include the target delivery role (of users who should receive the alert), alert name, and alert description. In the unescorted visitor awareness specification in Figure 3, escorted zone visits matching the criteria (of having a suitably long unescorted period) are output from the Unescorted Zone Visit (operator 7) and flow to Alert Delivery event operators 3 and 4, which generate alerts for unescorted visitors found in the facility, but to different roles. One operator alerts the security guard and the other operator alerts those people playing the situation-specific role of the escorts who are failing to escort their visitor.

The *Information Gathering* event operator issues tasks (via the Content Routing Engine) to security personnel and the VP Suite. User customization of the operator adds reference to a task specification and a role of users to who can perform the task. The special role "VP suite" is used to task the VP suite to run a video analysis algorithm. The input event stream to an Information Gathering operator effectively describes real-world events that indicate that more information is needed by VEAS. The task specification describes an information gathering task to be executed for each such event.

The Information Gathering event operator turns each of its input events, which describe a real world situation, into a corresponding request to gather needed information via an information gathering task, such as the running of video analysis algorithms or consulting security personnel for their assessments. The eventual execution of these information gathering tasks by the Content-based Routing Engine both consume resources and create delays. Thus, the Information Gathering event operator is a departure from the strictly incremental processing approach used by other event operators.

The results produced by information gathering tasks do not directly flow to the Awareness Engine. Instead, the information flows to the Awareness Engine indirectly via published changes in the Entity Model. There are two Information Gathering event operators along the top of the specification in Figure 3. Event operator 1 tasks the receptionist to identify each unknown person in the lobby. For visitors known to be in the facility, event operator 2 will generate an automatically executed task that will look through an expected visitor log, determine the appropriate escort(s) for the visitor, and tag the visitor identity attribute in the Entity Model with this information. This information is used by event operator 8 in determining whether the visitor is escorted.

6 CONTENT-BASED ROUTING ENGINE

The Content-based Routing Engine (CBRE) combines the coordination capabilities of workflow systems (Georgakopoulos, et al., 2000; BEA; TIBCO; Vitria; Georgakopoulos, 2004) with the content routing and syndication capabilities of existing content management systems (EMC; FatWire; FileNet; Georgakopoulos 2004). CBRE-provided workflow and content management capabilities are needed to dynamically service the following types of requests from the AE:

- prioritized alert requests for alerts that must be delivered to security personnel; and
- prioritized task requests for tasks that need to be performed for gathering more information.

To service prioritized alert requests, the CBRE supports content and alert *routing policies* that determine the appropriate user (or subset of users) to route specific alerts and related evidence (e.g., video segments). In addition, CBRE provides a mechanism and a client tool that allows users to view alerts, and drill down into the video evidence for each alert.

CBRE processes prioritized task requests by automatically invoking video analysis algorithms from the VP Suite, assigning tasks to users playing security personnel roles, and coordinating the execution of all tasks that are required by an awareness specification. These CBRE functions are discussed in more detail in the following sections.

6.1 Processing of Tasks

To support task assignment, coordination, and automation, the CBRE includes workflow management functionality. In particular, CBRE's workflow capabilities are similar to those provided by CMI (Georgakopoulos, et al., 2000).

Just like many other workflow management systems, CBRE's workflow component provides for the automation/enactment of workflow *processes.* Workflow processes typically consist of recursively defined activities, and specify control flow and dataflow between them. Basic activities, the leaves of the process specification tree, may be assigned to humans playing various roles, or be automatically executed using available computing resources.

The *Task Specifications* in Figure 2 are actually workflows specifications that have been defined by VEAS administrators to automate the tasking of video processing algorithms and security personnel. CBRE selects, instantiates, and enacts task specifications in response to requests from AE. Although CBRE's workflow component offers general workflow capabilities, the tasks, data flow, and control flow performed by CBRE (and VEAS in general) are typically centered on the orchestration of information gathering (involving VEAS video analysis and people). This permits pre-integration of video of the finite set of video analysis algorithms in the VP Suite, and the definition of typical human tasks for providing expert opinion and performing security-related activities (e.g., ask a person for an ID, secure a space in the facility, etc.) in a palette of reusable task specifications that require little or no customization and that can be readily utilized in awareness specifications.

Tasks to be executed are specified through the Information Gathering event operator, described in Section 5.3. In the example in Figure 3, the Awareness Engine issues a *Determine Identity* task request to CBRE to collect additional information. This task is pre-specified and implemented by a workflow processes consisting of several subtasks. The first subtask queries the Entity Model in the EEE to determine the time the person under investigation was in the restricted room. This is followed by a task that involves the execution of a face recognition algorithm on video collected during this time interval. If the face recognition algorithm fails to find at least one video frame with the person's face, a security guard is tasked to go to the restricted room and question the unidentified person as a last resort. Results of the *Determine Identity* flow back to the Entity Model via the Event Extraction Engine. If such a task determines the identity of the person and his presence in this room is a security threat (e.g., he is a visitor alone in a room that requires visitors to have escorts), the Awareness Engine will issue an alert request to the CBRE which then routes the alert to the specified security guard (e.g., the security guard who is closer to the visitor's current location).

6.2 Alert Delivery and Drill Down

Just like many other workflow engines (Georgakopoulos, et al., 2000; BEA; TIBCO; Vitria; Georgakopoulos, 2004), CBRE provides a *worklist* mechanism and a corresponding tool. Alerts are delivered in the worklist of VEAS users. Users may select a specified alert to view its properties, play video segments with evidence, and drill down to determine its causes and view related video evidence. This CBRE drill-down mechanism involves the display of details of how the triggering awareness specification decided the occurrence of a complex event, images and animations of entities moving within the facility, annotated or raw videos, and other relevant information.

Figure 4 shows an alert in the worklist tool in the backmost window. The next window shows the

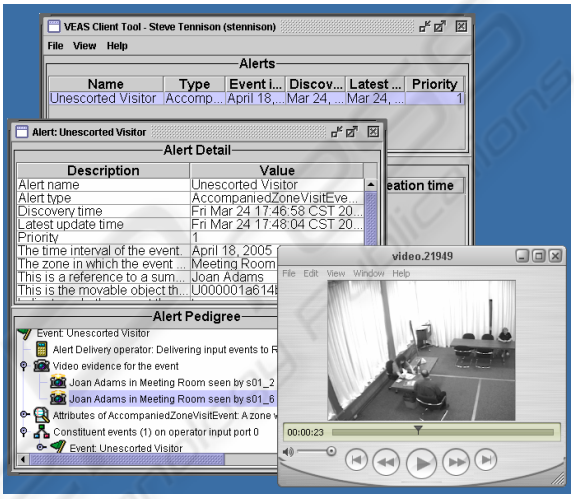

Figure 4: Alert drill down and video evidence.

detail of the unescorted visitor alert, which includes the pedigree and video evidence for the alert. The unescorted visitor is seen in the right of the video frame within a video segment that was given as evidence for the alert.

6.3 Routing Policies

The workflow component in the Content-based Routing Engine automatically routes alarms and tasks to specific roles. Therefore, the CBRE needs a mechanism to associate users and roles. This is accomplished by defining CBRE-supported *routing policies* that describe how specific users of VEAS correspond to the roles they might play in viewing specific security alerts and/or giving VEAS expert feedback as part of an information gathering task. Routing policies include:

- *Alarm delivery polices*: Definition of these policies involves the association of VEAS users with specific target roles (e.g., assign user "John Smith" to the role "security officer on duty in entrance B"). These policies are meaningful in a specific facility.

- *Expert feedback polices*: These associated VEAS users with "expert" and "security" roles (e.g., to review a video segment, secure a space, etc.).
- *Delegation and escalation polices*: These policies define what to do when an alert or a request has been delivered, but it cannot "consumed" by a user in the assigned role. Policies in this category permit a user to delegate an alarm directed to him/her to another role, and redirect alarms to other roles when a specified timeout expires.

7 RELATED WORK

The most advanced commercial surveillance systems, e.g., (GVI), utilize multi-camera motion detection to detect simple predefined events, and they are useful for guarding a perimeter (e.g., track potential intruders along a fence). Unlike VEAS, such surveillance systems cannot detect user–specified complex events, cannot utilize installation-specific knowledge (e.g., know who is a visitor or an employee, or be aware of spaces requiring escort and those that do not), and cannot combine and/or task sophisticated video analysis algorithms.

Early event processing systems, such as Snoop (Chakravarthy, 1994) developed *event algebra* based models, with generic event operators such a filter, sequence, and count. CEDMOS (Baker, et al., 1999) moved toward self-contained events and the computation of event parameters for complex events. Although these systems explored ideas that have been adopted by VEAS and stream databases, usability and efficiency were not addressed.

Stream databases, e.g., STREAM (Stanford University), Aurora (Abadi, et al., 2003), TinyDB (UC Berkeley), Borealis (Borealis Project; Abadi, et al., 2005), and Streambase (Streambase System), utilize a relational model with SQL enhanced with time-based windows for data streams. Operators are based on generic relational operators, i.e., selection, projection, join, aggregates, and group-by.

The VEAS model and language, which builds on ideas from our earlier awareness work (Baker et al., 2002), includes surveillance-specific operators are built on a dynamic Entity Model. They are specializations spatial operators (e.g., in Meeting room 2), temporal operators (e.g., workweek, holiday, 3rd shift), entity identification operators (e.g., visitor, employee in a specific organization), and relational algebra operators.

To perform event processing, stream databases (e.g., STREAM (Stanford University), Aurora (Abadi, et al., 2003), TinyDB (UC Berkeley), and Borealis (Borealis Project; Abadi, et al., 2005)) require use of time windows for stream queries and assume that there is no late arriving information. Optimization is performed assuming that input information is readily available and no information extraction cost is considered.

In contrast, VEAS requires no time windows (which is a requirement in video surveillance due to arbitrarily late arriving information). It performs incremental computation, and deals with information arriving late. These capabilities permit awareness specifications to take into account and reduce the cost of video analysis tasks.

Stream databases have no tasking and information gathering capabilities. In contrast, VEAS proactively gathers information that is missing to confirm or refute a partially matched event pattern within an awareness specification. VEAS is capable of tasking and executing video analysis algorithms and/or involve human (e.g., subject matter expert) to collect needed information and decision making.

The CBRE combines the coordination capabilities of workflow systems (Georgakopoulos, et al., 2000; BEA; TIBCO; Vitria; Georgakopoulos, 2004) with the content routing and syndication capabilities of existing content management systems (EMC; FatWire; FileNet; Georgakopoulos, 2004). In addition, CBRE provides a novel drill down capability for determining the evidence of and pedigree of each alert in support of situation understanding by end-users, as shown in Figure 4.

8 CONCLUSION

VEAS helps automate surveillance by analyzing surveillance video from potentially thousands of cameras and other non-video sensors and *automatically* detecting complex events that indicate situations of interest, *alerting* humans about them, providing the *evidence* that led to the alerts, and do this in near *real-time*—at pace with their input video streams. When information is missing or uncertain, VEAS has the capability to gather additional information proactively to make the appropriate determinations. In this paper, we focused in the presentation of VEAS's novel runtime architecture and event processing capabilities, and described the application of these in the video surveillance domain. The novel capabilities and key benefits of

VEAS are increased alert accuracy (by narrowing the video analysis problem via installation-specific customization), proactive gathering of additional information, and the ability to *monitor* any facility of interest, and to do so *inexpensively* and in *near real-time*. VEAS is *scalable* and does not require adding more humans to monitor the additional security cameras when a facility is expanded.

Although in this paper we focused on video surveillance, VEAS can effectively utilize non-video sensors and/or be applied in other domains. Our directions for future work include applying VEAS to support complex event detection in domains related to meetings and conferences. We are also working to extend VEAS capabilities for managing non-video sensors, such as RFID readers, badge scanners, and intelligent locks aimed at providing Critical Infrastructure Protection (CIP) without relying on video. We are also experimenting with algorithms for learning of normal event patterns to enable VEAS to automatically detect anomalous events that have not been anticipated.

REFERENCES

Abadi, D. J., D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik (2003). Aurora: a new model and architecture for data stream management. *VLDB Journal,* 12(2).

Abadi, D. J., Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik (2005). The Design of the Borealis Stream Processing Engine**.** *Second Biennial Conference on Innovative Data Systems Research* (*DR 2005)*.

Aref, W., A. C. Catlin, A. Elmagarmid, J. Fan, J. Guo, M. Hammad, I. Ilyas, M. Marzouk, S. Prabhakar, A. Rezgui, S. Teoh, E. Terzi, Y. Tu, A. Vakali and X. Zhu (2002). A distributed server for continuous media. *18th International Conference on Data Engineering* (ICDE'02).

VACE (2006). VACE Phase III BAA Discussion. Retrieved 2006 from http://www.nbc.gov/fort_h /vaceiii/BidderBrief-final.ppt.

Baker, D., D. Georgakopoulos, H. Schuster, and A. Cichocki (2002). Customized Process and Situation Awareness. *International Journal of Cooperative Information Systems*, 11(3 & 4), World Scientific.

Baker D., A. Cassandra, and M. Rashid (1999). CEDMOS: Complex Event Detection and Monitoring System. *MCC Technical Report CEDMOS-002-99*. Retrieved 2005 from http://citeseer.ist.psu.edu/ baker99cedmos.html

BEA (nd). WebLogic. Retrieved 2005 from http://www.bea.com/.

Borealis Project. Borealis Second Generation Stream Processing Engine. Retrieved 2005 from http://nms.lcs.mit.edu/projects/borealis/.

Chakravarthy, S. (1994). Snoop: An Expressive Event Specification Language for Active Databases. IEEE *Data and Knowledge Engineering*, 14(10).

EMC (nd). Documentum. Retrieved 2005 from http://software.emc.com/.

FatWire (nd). Content Server. Retrieved 2005 from http://www.fatwire.com/.

FileNet (nd). Content and Workflow Managers. Retrieved 2005 from http://www.filenet.com/.

Georgakopoulos, D., H. Schuster, D. Baker, and A. Cichocki (2000). Managing Escalation of Collaboration Processes in Crisis Mitigation Situations. *16th International Conference on Data Engineering* (ICDE'00).

Georgakopoulos, D. (2004). Teamware: An Evaluation of Key Technologies and Open Problems. *Distributed and Parallel Databases*, 15(1).

GVI (nd). GVI TRAK. Retrieved 2005 from http://www.samsung-security.com/.

Hongeng, S., R. Nevatia, and F. Bremond (2003). Video-based Event Recognition: Activity Representation and Probabilistic Recognition Methods. In *Computer Vision and Image Understanding*, 96.

Moser, F., A. Kraiss, and W. L. Klas (1995). A buffer management strategy for interactive continuous data flows in a multimedia DBMS. *21st International Conference on Very Large Data* Bases (VLDB'95).

Nodine, M., J. Fowler, T. Ksiezyk, B. Perry, M. Taylor and A. Unruh (2000). Active Information Gathering in InfoSleuth. *International Journal of Cooperative Information Systems,* 9(1/2).

Stanford University (nd). STREAM: The Stanford Stream Data Manager. Retrieved 2005 from http://www-db.stanford.edu/stream/.

Streambase (nd). Streambase System. Retrieved 2005 from http://www.streambase.com/.

TIBCO (nd). TIBCO Staffware Process Suite. Retrieved 2006 from http://www.tibco.com.

UC Berkeley (nd). TinyDB: A Declarative Database for Sensor Networks. Retrieved 2005 from http://telegraph.cs.berkeley.edu/tinydb/.

Vitria (nd). BusinessWare. Retrieved 2005 from http://www.vitria.com/.