Forensic Video Reconstruction

Larry Huston[†]
larry.huston@intel.com

Rahul Sukthankar[†‡]
rahul.sukthankar@intel.com

Jason Campbell[†‡]
jason.d.campbell@intel.com

Padmanabhan Pillai[†]
padmanabhan.s.pillai@intel.com

[†]Intel Research Pittsburgh
417 S. Craig Street Suite 300
Pittsburgh, PA 15213
U.S.A.

[‡]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.

ABSTRACT

This paper describes an application that enables quick reconstruction of interconnected events, sparsely captured by one or more surveillance cameras. Unlike related efforts, our approach does not require indexing, advance knowledge of potential search criteria, nor a solution to the generalized object-recognition problem. Instead, we strategically pair the intelligence and skill of a human investigator with the speed and flexibility of a parallel image search engine that exploits local storage and processing capabilities distributed across large collections of video recording devices. The result is a system for fast, interactive, brute-force video searching which is both effective and highly scalable.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*; C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems; I.4.9 [**Computing Methodologies**]: Image Processing and Computer Vision—*Applications*

General Terms

Algorithms, Human Factors, Performance

Keywords

video retrieval, active storage, interactive search

1. INTRODUCTION

As surveillance cameras proliferate, the resulting glut of video poses a crippling data interpretation challenge. Present approaches to surveillance video analysis rely on linear searches performed by human beings to interpret and correlate observed activity between multiple sources. This technique rapidly becomes intractable as the number of deployed cameras increases. For forensic reconstruction of events, such as crime scenes, these approaches will always be too slow to provide real-time guidance to investigators in a rapidly-developing situation.

Researchers in a number of fields are pursuing a variety of partial solutions to the above problems; many commercial surveillance video recorders utilize extreme temporal and data compression to reduce storage requirements and search time, but in doing so discard potentially important video frames from the outset and provide no alternative to linear human search. Sensor network approaches [9, 12, 13] address the question of adapting a collection of data gathering nodes or cameras to look for particular patterns, but are *continuous query* systems which do not archive raw source data (video) nor provide facilities for *ex post* searching except as anticipated by prior queries. Content-based image retrieval [2, 5] and object detection techniques [18, 20] offer a number of potentially interesting domain-specific classifiers and recognizers, though no algorithm offers a general resolution to the object recognition problem, and most algorithms are too computationally expensive to run on every frame of surveillance video (e.g., the Blobworld [2] image segmentation algorithm takes more than a minute to process a single frame on modern hardware). Image indexing in general, whether indices are built by machines or by humans, can offer fast response to queries, but only by increasing the cost of video acquisition and only when potential query criteria can be identified in advance. Recent research in video surveillance and monitoring (VSAM) [3,7] addresses online image understanding, object recognition, and tracking. In this paper we focus instead on searching raw video data. Any advances that add semantic information to the video data as it is captured are complementary to our approach, and this additional information has the potential to make our searches more efficient and fruitful.

We propose a new approach to surveillance video analysis which combines the interactive use of a human investigator's skill with the speed and flexibility offered by the automated, highly parallel, brute-force application of image processing techniques. *Interaction with a human investigator* is essential because fully-automatic extraction of semantic content from images remains significantly inferior to human performance, despite decades of research in computer vision and image processing. For the foreseeable future, the user should be a key component of any forensic video reconstruc-

VSSN'04, October 15, 2004, New York, New York, USA.

tion system. Automatic image processing could best be used to eliminate clearly irrelevant data, reducing the amount of information presented to the user, and utilizing the human investigator's limited attention more effectively. *Large-scale parallelism* is required in order to permit quick searching of a large and growing number of cameras and to enable a distributed search to execute close to data storage locations where it will be unhindered by WAN/MAN/Wireless bandwidth limitations. Finally, *brute-force search* (as opposed to indexing) is unavoidable in surveillance video analysis because search criteria will generally not be known until after an incident has been recorded. We have implemented our approach in VideoFerret, an application that leverages recent work in automated, highly parallel, brute-force image search [10]. This is integrated into a user interface that emphasizes flexibility, concurrency, fast response times, early presentation of partial results, and spatial and temporal visualization techniques.

This paper's novelty lies not in its image processing algorithms, nor in any efficient scheme for indexing the data, but rather in the ability to perform user-defined queries on unstructured surveillance data and to visualize events through time and across multiple cameras. Our approach to the forensic video reconstruction problem enables both human and computer resources to tackle aspects that are best suited to their abilities: the user can focus on interpreting and correlating semantic information while the computer tackles the tedious task of search and organizing large volumes of raw data.

The remainder of this paper is organized as follows. Section 2 explains the general use scenario which has motivated the design of VideoFerret. Section 3 casts forensic video reconstruction as a distributed video search problem and describes our approach more formally. Section 4 describes our forensic video reconstruction application in more detail. Section 5 describes the underlying implementation. Section 6 presents preliminary results. Section 7 concludes the paper.

2. USAGE SCENARIO

Consider the following scenario: A masked gunman commits a violent bank robbery and escapes in an unknown getaway car before the police arrive on the scene. The crime was recorded on the bank's security cameras and nearby public areas were photographed by numerous surveillance cameras. The police would like to understand what took place by analyzing this data — hopefully quickly enough to predict probable current locations of the suspect.

Using current approaches, all of the data would be gathered in a centralized location and manually searched. Investigators would first collect video from cameras which may have recorded relevant material — a process which might itself take days — and then view each video to attempt to manually reconstruct the chronology of events surrounding the crime. Even with accelerated video playback this process will never be quick enough to offer real-time guidance to police immediately following a crime.

This paper presents a system that enables investigators to search surveillance video using real-time, interactive, content-based image retrieval techniques which execute in parallel across numerous active video storage devices (each connected to one or more associated cameras). This type of adaptive search provides two advantages: First, it makes more efficient use of an investigator's time by leveraging computer vision techniques and highly parallel brute-force search to eliminate irrelevant data, enabling a more thorough investigation in the available time. Second, it allows more timely access to relevant portions of the surveillance data, which can assist police in responding quickly during unfolding events.

Using our application, VideoFerret, the investigator scans the video taken at the bank and selects a few images that show the masked robber. These images are used to build a query for the particular clothing (color and visual texture) worn by the suspect, and this query is sent to *active storage devices* [1, 11, 15] attached to surveillance cameras. Each of these devices searches its recent history of digital images and video for people whose clothing matches the query. Several respond quickly and the investigator browses the partial results, rejecting those that are unlikely to be the suspect. One of the videos fortuitously shows the suspect getting into an escape vehicle. The investigator is able to create a second, concurrent query for vehicles of this type. Another camera spots the same vehicle running a red light several miles away, and this enables the investigator to focus the search on a particular neighborhood. Meanwhile, the clothing search has returned thousands of hits in the city. By focusing on the hits in the selected neighborhood, the investigator is able to reduce the number to a manageable size. Several of the people appear similar to the suspect in size and build (an evaluation made by the investigator, as it is too sophisticated for the system's image processing algorithms). Some of the hits also match the vehicle profile, and further manual examination shows that a camera has spotted the suspect alighting from the getaway car. The investigator aborts the extraneous searches and uses VideoFerret to track this individual forward through time. Soon, the investigator feels sufficiently confident to send police vehicles to the suspect's predicted location.

3. DISTRIBUTED VIDEO SEARCH

The forensic video reconstruction problem can be formulated as follows. A large number of geographically-distributed sensing nodes collect and store images. Each sensing node consists of a camera, a processor for executing searches, persistent storage, and a network interface. The investigator is equipped with a computer that runs the VideoFerret application and connects to the sensing nodes (see Figure 1). The queries posed to the system cannot be specified *a priori* because they will depend on specific details relevant to the crime scene. For example, for one crime scene, investigators may wish to search for a specific car; for another, they may wish to find people loitering near an ATM.

Traditional image retrieval approaches pre-process data and build indices. This is impractical for forensic video reconstruction for several reasons. First, many algorithms cannot keep pace with the rate at which new data is generated. Second, the majority of images will never be searched before they expire, so the effort involved in pre-processing this data is wasted. Third, the high dimensionality of the data and lack of *a priori* knowledge about the query precludes the use of many indexing schemes.

To address these issues, we employ brute-force search [10] of the raw data in conjunction with any semantic attributes that can be computed as the data is acquired. Brute-force search has received little attention because of its perceived impracticality for performance reasons. We have built an infrastructure, Diamond, that addresses many performance challenges associated with brute-force search. Diamond leverages the natural parallelism of the search problem by executing application-specific search code at each storage device instead of processing the data at a centralized site. This is detailed in Section 5.2.

We believe that forensic video reconstruction searches should be interactive because fully-automatic extraction of semantic content from images remains unattainable with the current state of the art in computer vision and image processing. For the foreseeable future, the user should be a key component of any forensic video reconstruction system. However, image processing can be used to screen out clearly irrelevant data, reducing the amount of information pre-

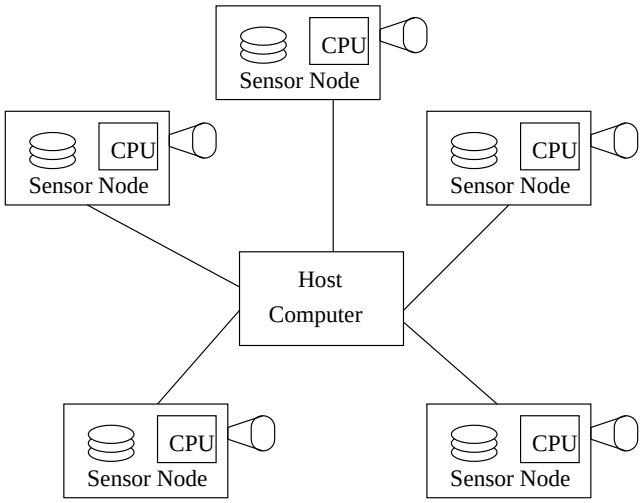

Figure 1: System Overview - the investigator executes the search on the host computer, which is connected to a large set of active sensor nodes over a WAN. Each sensor node is connected to one or more surveillance cameras and is equipped with local storage and processing.

sented to the user, and utilizing the user's limited attention more effectively.

While Diamond dramatically accelerates brute-force search, most realistic searches will not complete within seconds due to the large volume of data. Since the user's attention is a precious resource, we have designed VideoFerret to utilize this resource as effectively as possible. Some of these criteria are summarized below.

- **Multiple Concurrent Searches:** A VideoFerret user can initiate additional searches based on the partial results from an initial search, and yet continue to pursue the first search as well. For instance, an image of the suspect meeting with accomplices could be used to define additional searches that look for those individuals as well. This is important for three reasons. First, it allows the investigator to pursue new inquiries without losing focus on the current chain of events. Second, by starting an additional search as soon as possible, results could be available by the time the investigator is ready to focus attention on the new topic. Third, the investigator is able to simultaneously test multiple hypotheses by creating several concurrent searches.

- **Useful Partial Search Results:** VideoFerret presents partial search results to the user as they become available. This provides several advantages. First, the user is able to determine that a particular query is fruitless (e.g., due to an excess of false positives) without waiting for all of the data to be processed. Second, the user may find sufficient evidence from the partial results to suspend the search (e.g., the investigator may find a clear image of the getaway car's license plate early in the search). Third, partial results can be used as input for additional searches. Other research on queries over large databases has also argued for presenting partial results to the user [8].

- **Special-Purpose Visualizations:** VideoFerret provides visualization interfaces to enable the investigator to organize search results both spatially and temporally, to facilitate reasoning about events at the crime scene. For instance, knowing that the getaway car left the bank at a particular time allows the investigator to discard matches for this vehicle at the same time from distant locations. It also enables the investigator to focus on images taken by cameras at a particular location. When showing matches, VideoFerret can highlight the regions in the image that satisfy the search criteria, helping the investigator to understand why a particular image was returned and fine tune the search.

- **Interactive Search Parameter Adjustment:** VideoFerret allows the investigator to interactively adjust search algorithm parameters (e.g., distance metrics and thresholds) to select the appropriate tradeoffs between precision and recall. For instance, for cameras in challenging lighting conditions, the investigator may choose to use lower thresholds and accept many false positives in exchange for a chance at catching a glimpse of the criminal. VideoFerret also provides visual feedback to help the investigator explore the impact of changing parameters on sample images.

These criteria enable the user to focus attention on understanding the semantic content of the video, while exploiting the computer's ability to perform brute-force seach across large collections of data. The following section describes the interface that we have implemented in VideoFerret to address these issues.

4. VIDEOFERRET INTERFACE

This section describes the interface for VideoFerret, the forensic video reconstruction search application that runs on Diamond. As described above, a key requirement in VideoFerret's design is to maximize the use of the investigator's attention span. Additionally, we have the following requirements for our search application: an investigator should be able to define searches by providing sample image patches, VideoFerret should provide a useful set of content-based search algorithms, and it should be easy to extend VideoFerret by adding new image retrieval algorithms.

Figure 2 shows a screenshot of VideoFerret. The interface is divided into three regions. The *camera display* enables the investigator to select a particular camera and manually browse its recorded video stream. The *search display* organizes the current set of concurrent searches. Each search allows the investigator to view the hits matching the particular query and organize the results by camera, time, etc. The *map display* highlights, for each hit, the corresponding camera location. This helps the investigator to visualize events at the crime scene, and track them across space and time. The following subsections describe each of these components in greater detail and how their design meets our application requirements.

4.1 Camera display

The camera display allows the investigator to browse recorded video from one or more selected cameras (see Figure 3). The display supports toggling between multiple camera views to provide different viewpoints of the same event. The investigator can easily jump to a particular time, or manually browse forward and backward in time looking for clues. This is particularly valuable for cases when the investigator cannot easily specify a query suitable for content-based image retrieval algorithms. For instance, a suspect may have deposited a stolen item in a trash can for his accomplice. The investigator would like to track the person who retrieved the stolen item. Ideally, one would like to automatically find the images containing the stolen item. Unfortunately, this may be challenging because of poor camera resolution, occlusion, or an

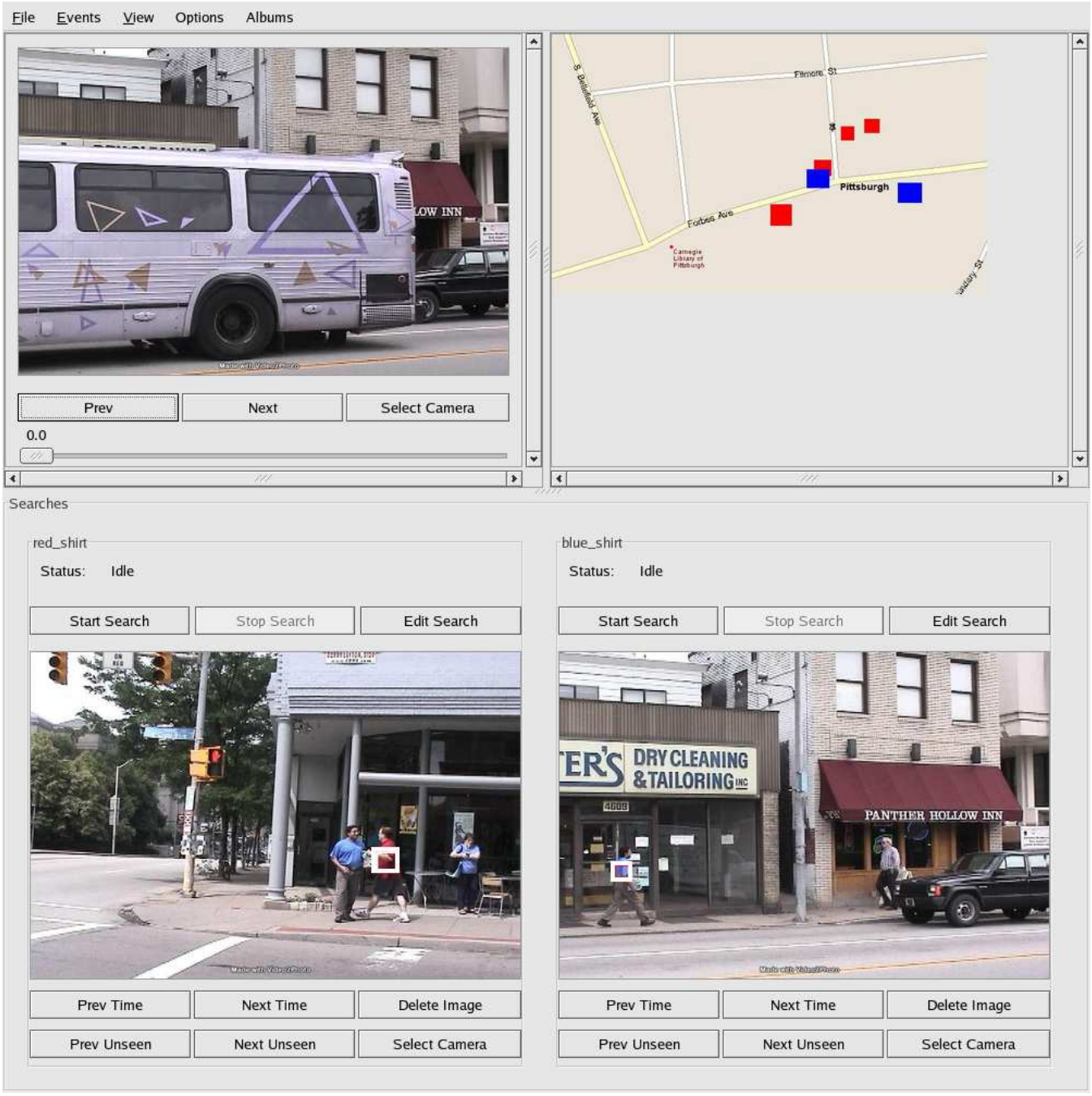

Figure 2: VideoFerret Screenshot - This figure presents the main window of the forensic video reconstruction application. The camera view (upper left), displays video from a selected camera. The map view (upper right) shows the physical locations of the cameras where images matching the search criteria were taken. The lower half of the screen contains a region for each of two active searches and displays individual images that match each of the queries.

Figure 3: Camera Display - A screenshot of the VideoFerret camera display. This display allows the investigator to browse data captured by a specific camera.

object's nondescript or deformable appearance. Using the camera display, the investigator can manually identify key frames where "interesting" events occurred.

Once frames containing objects of interest are found, the investigator can create a new search using the identified objects as examples to the content-based image retrieval algorithms. For example, the investigator may highlight the suspect's plaid shirt and initiate a color histogram search to find regions in other images with similar color distributions.

4.2 Search display

The search display enables the investigator to manage and view results from multiple concurrent searches. Each search is allocated a separate area of the display with its own set of controls (see Figure 4.

Search management allows the user to start or abort a current search, or to refine the current search criteria. A search consists of a boolean combination of predicates that are used to identify images of interest. Each predicate consists of an algorithm and the configuration state for that particular algorithm, such as parameter settings (e.g., threshold, window size, distance metric) and training exemplars (e.g., color or texture patches). VideoFerret currently supports a variety of image retrieval algorithms based on color, texture and shape features, including trained object detectors (described in Section 5.3.1); it also supports queries based on non-image metadata such as time, geographic location or camera-id. The investigator can edit a search by adding new predicates, or changing the parameters and exemplars for existing predicates. When editing these parameters, the investigator can visualize the effects of these refinements on a selected set of images (see Figure 5). This allows the investigator to specialize the search for the desired object, and to appropriately tune each predicate's precision/recall tradeoff.

The search display also provides interfaces for organizing results from currently-executing searches. These partial results can be displayed using several views: grouped by camera, sorted by time, or arranged by whether they have previously been seen. As results are displayed, regions that match the selected predicates are highlighted in the image. As the investigator browses images, false

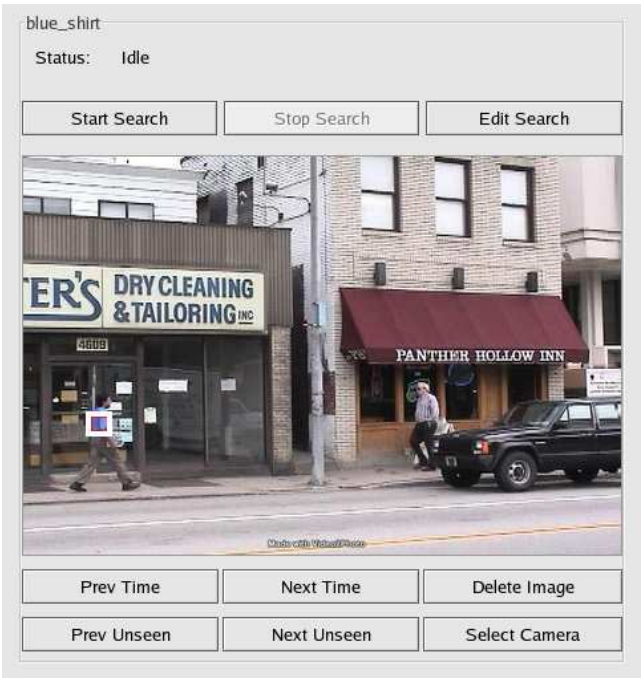

Figure 4: Search Display - This screenshot shows an active search. The video frame displayed is one of the images retrieved by this search. The various buttons allow the investigator to reconfigure the search and to control which images are displayed.

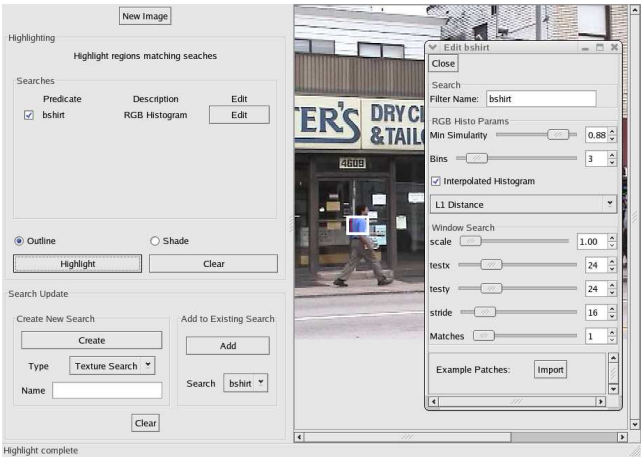

Figure 5: Refinement Window - The investigator uses this window to adjust search parameters. Given a sample image, the investigator can adjust the current search parameters and see which regions match the adjusted predicate. In this case, the image region marked with a white rectangle matches the current search parameters.

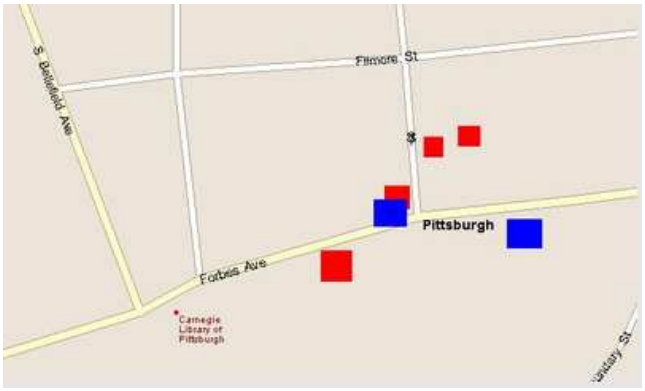

Figure 6: Map Display - This window shows where images have been found for the different searches. Each color indicates a different search. The marked regions correspond to the approximate area viewed by each of the matching images / camera views.

positives can be tagged and discarded. Similarly, the investigator can select regions from these images as additional exemplars for predicates in the current or other searches.

4.3 Map display

The map display (see Figure 6) shows where potential matches were found in both time and space. When an match is found in one of the active searches, the area on the map corresponding to the camera is marked. The investigator can control which searches are displayed as well as selecting time ranges. This allows the investigator to visualize the time-based flow of the object of interest across the different cameras.

The investigator uses the map display to see where potential hits are found. Using this map, events that are false positives can be eliminated based on additional domain information known to the investigator. For example, knowing that the suspect left the crime scene on foot allows the investigator to safely eliminate potential matches that find the suspect across town one minute later.

5. SYSTEM IMPLEMENTATION

The forensic video reconstruction system described in this paper consists of three main components: (1) active sensor nodes, which collect and store video data from the video cameras; (2) the Diamond infrastructure, which enables distributed search over the stored data; (3) the VideoFerret application, which encapsulates the the content-based image retrieval algorithms and which provides the user interface. The following sections examine these three components in more detail.

5.1 Active Sensor Nodes

The active sensor nodes consist of a standard computer connected to one or more video cameras. These nodes are equipped with local storage to archive sensor data. Rather than simply storing the raw images from each camera, VideoFerret employs Multi-Fidelity Storage (MFS) [14] to adaptively compress the incoming video stream in an application-dependent manner. This allows the system to more effectively utilize the limited storage capacity at each node. In particular, each active sensor node may elect to drop (or store at reduced fidelity) images based on an application-specific interest metrics. For crime-scene detection, VideoFerret may prioritize storage for those frames containing human faces and

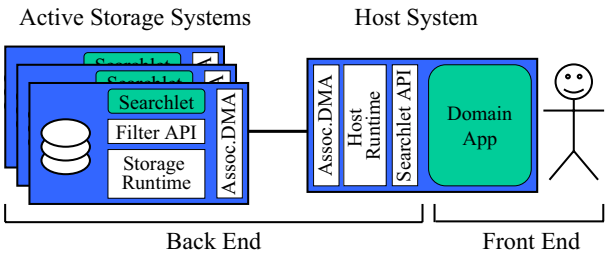

Figure 7: Diamond Architecture - The front end encapsulates the domain-specific application code and the back end provides a generic search infrastructure.

traffic activity. The sensor node is network-accessible and exports an interface that permits remote searching, as described in the next section.

5.2 Diamond

VideoFerret uses the Diamond system [10] to efficiently perform brute-force search across the distributed sensing nodes. The Diamond architecture (see Figure 7) separates the *front end*, which encapsulates domain-specific application code, from the *back end*, which consists of a domain-independent infrastructure. By separating these two components we hope to build search infrastructure that can be used for a wide range of search applications.

Diamond uses active storage [1,11,15], where processing is coupled with storage, to enable application-specific code to be executed locally on the stored data. Remote execution allows the search to be parallelized providing more aggregate computation power than in a centralized search. Executing search code near where the data is stored also eliminates the cost of transferring the data to a centralized site; this is important in the forensic video reconstruction scenario, where the different cameras may be connected over a wide-area network or wireless infrastructure. Finally, as cameras (and corresponding active storage devices) are added to a surveillance network, the total computing power available in the network increases appropriately.

A Diamond application runs on the host computer and interacts with the user to formulate a query. Once a query has been formulated, the application translates the query into a set of machine executable tasks (termed a *searchlet*) that the back end uses to discard data that does not match the search criteria. The searchlet contains the domain-specific knowledge necessary to filter out hopeless data and acts as a proxy of the application (and of the user) that executes within the active storage devices at each sensing node.

A searchlet consists of a set of *filters* and some configuration state (e.g., filter parameters and dependencies between filters). For example, a searchlet to retrieve portraits of people in dark business suits might contain two filters: a color histogram filter that finds dark regions and an object detector that locates human faces. Each of these filters can independently discard an image. Images that pass through all filters in a searchlet are deemed interesting, and made available to the domain application through the searchlet API.

Diamond is designed for interactive search and exploits several simplifications inherent to the search domain. First, search tasks only require read access to data, allowing Diamond to avoid locking complexities and to ignore some security issues. Second, search tasks typically permit stored images to be examined in any order. This order-independence offers several benefits: easy parallelization within and across storage devices, significant flexibility

in scheduling data reads, and simplified migration of computation between the active storage devices and host computer. Third, most search tasks do not require maintaining state between images. This "stateless" property supports efficient parallelization and simplifies the run-time migration of computation between active storage device and host computer.

The Diamond system handles many of the complexities of a distributed system; communicating with multiple devices, optimizing the order the filters are evaluated, and dynamically partitioning computation between the host and storage devices. This separation allows the application writer to focus on algorithms for searching the images.

The domain application may perform further processing on the interesting images to see if they satisfy the user's request. Because such processing can be carried out centrally, at the application host CPU, this additional processing can be more general than the processing performed at the searchlet level. For instance, the additional processing may include cross-image correlations and querying auxiliary databases. Once the domain application determines that particular image matches the user's criteria, that image is displayed to the user. When processing a large data set, it is important to present the user with results as soon as they appear. Based on these partial results, the user may choose to refine the query and restart the search. Query refinement leads to the generation of a new searchlet, which is once again executed by the back end.

To increase interactive performance, Diamond caches the results of each filter to help answer subsequent queries. This caching is implemented by uniquely identifying each filter and its dependencies (attributes and arguments) without explicit support from the application or the application developer. The cache provides two primary benefits for improving interactive search time. First, the cache reduces the search space by quickly eliminating objects that will not pass a filter that has been previously executed. Second, the cache reduces the amount of computation per object by storing previously computed results (*e.g.* feature extraction) instead of recomputing them for each search.

5.3 VideoFerret Implementation

Our forensic video reconstruction application is a flexible, extensible, application that supports a variety of content-based image retrieval algorithms, detailed in Section 5.3.1. The process for specifying a search is described in Section 5.3.2, and our approach for interactive refinement of searches is given in Section 5.3.3.

5.3.1 Image Retrieval Algorithms

VideoFerret currently supports region- and object-based image searches. All of these algorithms look for regions that contain the specified features instead of looking at the image as a whole.

Color filters

VideoFerret supports color-based searches using histograms [6, 19]. Unlike prior approaches to color-based image retrieval [4], where a fixed color representation was required for efficient indexing, VideoFerret offers the user the flexibility to interactively adjust the internal representation (e.g., number of bins, color spaces), region-based search parameters (e.g., the search steps in both scale and space) and similarity metrics (e.g., Manhattan, Euclidean or Earth Mover's Distance [17]). The effect of changing these parameters can be immediately visualized in VideoFerret (see Figure 5). The user can specify target histograms that match the desired concept by selecting patches from other images, such as those retrieved in earlier searches. This is a significant improvement over systems that require the user to select colors from a palette because the target histogram can be multimodal, representing multiple colors in the appropriate proportions (e.g., a plaid shirt). However, since color histograms cannot model the effects of variations in appearance due to illumination and viewpoint effects, the user may need to add several examples of color patches for certain concepts (e.g., water).

Texture filters

VideoFerret's interface for region-based texture filters is similar to the color filter described above. The user can build a customized filter for the desired visual texture by providing a few examples. The frequency content of the texture patch is represented using standard techniques (e.g., using a Laplacian Pyramid [6]). As with the histograms, the user can interactively adjust the search parameters and see their effect on some sample images.

Offline-trained Object Detection

VideoFerret also employs offline-trained object detectors, such as face detection algorithms [16, 18, 20]. These classifiers are best suited for identifying semantic content that is not user-specific, because they typically must be trained on large databases of positive and negative examples using machine learning algorithms. One drawback to these algorithms is that they typically provide users with little ability to adjust their performance (their trade-off between false-positive and false-negative rate is typically determined at training time). VideoFerret can mitigate this to some degree by supporting multiple classifiers for the same concept, each trained for a particular choice of running time and classification accuracy.

5.3.2 Specifying a Search

When starting a new search, the investigator must map his/her semantic request onto available algorithms. This is done by selecting one or more of the retrieval algorithms described in the previous section. Once an algorithm is identified, the investigator creates a predicate using that algorithm and sets its appropriate parameters. For most algorithms these include thresholds, window size, stride, scale, etc.; setting these parameters can enable the user to trade speed for search accuracy.

For an example-based algorithm (e.g., color or texture matching), the investigator must provide some sample image patches. The investigator may select such examples by highlighting relevant regions in the camera display or other image source.

Once the user has defined some predicates, he/she constructs a search by combining these predicates using boolean operators. A search returns all pictures that pass these predicates (e.g., combining a red and blue histogram will return images that contain both red and blue regions). When the investigator starts a search, VideoFerret generates a searchlet that corresponds to the defined search parameters and passes it to the Diamond system. Diamond evaluates this searchlet on each object in the data set and returns those objects that are flagged by the searchlet to VideoFerret for display to the user.

5.3.3 Search Refinement

As discussed earlier, a important feature of VideoFerret is the ability to interactively refine a search based on partial results. While a VideoFerret search is running, matching images are presented to the user as they become available. The user can look at these images and decide whether the current query is producing the desired results. If it is, then the user can scan the results until the desired images are located. A more likely scenario is the results do not precisely match what the user had in mind. In this case the user may refine the search by adding predicates or adjusting parameters. A

Search	Cameras	Avg Time (secs)	Std. Dev.	Transfer Time (secs)
S1	1	81.70	0.33	291
S1	4	84.87	2.12	1164
S1	8	85.01	3.93	2328
S1	12	84.67	0.33	3492
S2	1	81.74	2.59	291
S2	4	83.85	0.96	1164
S2	8	82.20	0.89	2328
S2	12	83.04	0.51	3492

Table 1: Search times - This table presents the average time needed to search archived video frames on a varying number of cameras (and corresponding active storage devices). Each camera stored 1028 video frames. We report the average time and standard deviation over 3 runs. Search S1 looked for images with a specific color distribution and a human face and search S2 looked for two different color distributions in the same image. The last column is estimated time to move the data to a centralized place over a 10Mbps network.

key to successful refinement is providing the user with visual feedback on how any potential changes will affect the results.

When the results are returned to the user, VideoFerret highlights the regions of the image that match the current query. This provides feedback on why the current images match. Additionally, VideoFerret allows the user to evaluate different predicates on sample images (returned results or from other sources) and visually highlight those image regions that match the current predicate settings. The user can then refine the predicates so that the search can include or omit specific images.

6. PRELIMINARY RESULTS

We have performed some preliminary experiments to validate the feasibility of our approach for searching archived surveillance video. In these experiments we executed searches on a varying number of cameras to see if we could search data in a reasonable time frame and to demonstrate that our approach scales with the number of camera added.

To get data for these experiments, we logged video from a camera at a rate of 1 frame per second (640x480 resolution) for approximately 17 minutes (1028 frames) and then replicated this data on each of the camera nodes. In these experiments our camera nodes contained 1.2 GHz Intel® Pentium® III processors with 512 MB RAM and 73 GB SCSI disks and the host system contained a 3.06 GHz Intel® Pentium® Xeon™ processor, 2 GB RAM, and a 120 GB IDE disk. The host and the camera nodes were connected using a 10 Mbps Ethernet.

Using this test setup we executed two different searches while varying the number of cameras involved in the search. The first search (S1) looked for a specific color distribution and a human face. The second search (S2) looked for two different color distributions to occur in the same image. We executed each search over the selected cameras and measured the time required to the search all the data.[1] Table 1 reports the average completion time and standard deviation over three runs of the search.

These results show that each search takes around 81–85 seconds. Both searches take approximately the same time because they are limited by the time needed to read the data off the disk and not by the computation (we expect to improve on this in later versions of Diamond). We also see that adding more cameras to the search does not significantly change the time needed for each of the searches.

In contrast, the last column of Table 1 shows the time necessary to transfer all the images to a centralized site using a 10Mbps network (we assume the data is compressed but we do not reduce the resolution). This time is the lower bound for searching the video at a centralized site. As expected, performing the computation where the data is stored offers significant performance advantages and these benefits become more pronounced as the number of cameras increases.

[1] In practice the time until the first result is delivered is a more practical metric since it reflects the user's idle time; however, accurately measuring this requires large numbers of real queries and data.

7. CONCLUSION

A lack of good *ad hoc* video search techniques precludes real-time analysis of historical and current data in large-scale surveillance systems. Linear human search is too slow, and automated search is typically inadequate due to the absence of a general solution to the object recognition problem. Indexing and attribute-based approaches fall short because data volumes are so high, potential interesting characteristics so varied, and search criteria cannot be known in advance.

We have described in this paper a new approach to surveillance video analysis: VideoFerret blends human strengths in choosing search strategies with highly parallelized automated image search capabilities. Key to our approach is the use of active storage devices associated with each surveillance camera. These active storage devices can quickly eliminate clearly irrelevant material during a search and thus reduce communication bottlenecks and simplify scaling. VideoFerret further speeds the interpretation of results by including facilities which help an investigator visualize search results in temporal, geographic, and image context. VideoFerret offers a system for fast, interactive, brute-force video searching which is effective, highly scalable, and can permit surveillance records to be useful during a response to an incident rather than days or weeks after the fact.

8. REFERENCES

[1] A. Acharya, M. Uysal, and J. Saltz. Active disks: Programming model, algorithms and evaluation. In *Proceedings of ASPLOS*, 1998.

[2] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8), 2002.

[3] R. Collins, A. Lipton, and T. Kanade. A system for video surveillance and monitoring. In *Proceedings of the American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems*, 1999.

[4] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4), 1994.

[5] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the QBIC system. *IEEE Computer*, 28, 1995.

[6] D. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall, 2002.

[7] W. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site.

In *Proceedings of IEEE Computer Vision and Pattern Recognition*, 1998.

[8] J. Hellerstein, R. Avnur, A. Chou, C. Hidber, V. Raman, T. Roth, and P. Haas. Interactive data analysis: The CONTROL project. *IEEE Computer*, August 1999.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[10] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. R. Ganger, E. Riedel, and A. Ailamaki. Diamond: A storage architecture for early discard in interactive search. In *Proceedings of USENIX Conference on File and Storage Technologies (FAST)*, 2004.

[11] K. Keeton, D. Patterson, and J. Hellerstein. A case for intelligent disks (IDISKs). *SIGMOD Record*, 27(3), 1998.

[12] R. Min, M. Bhardwaj, S. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan. Low-power wireless sensor networks. In *Proceedings of VLSI Design*, 2001.

[13] S. Nath, A. Deshpande, Y. Ke, P. Gibbons, B. Karp, and S. Seshan. Irisnet: An architecture for internet-scale sensing services. In *Proceedings of Conference on Very Large Data Bases*, 2003.

[14] P. Pillai, Y. Ke, and J. Campbell. Multi-fi delity storage. In *Proceedings of ACM Workshop on Visual Surveillance and Sensor Networks*, 2004.

[15] E. Riedel, G. Gibson, and C. Faloutsos. Active storage for large-scale data mining and multimedia. In *Proceedings of Conference on Very Large Data Bases*, August 1998.

[16] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1), 1998.

[17] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2), 2000.

[18] H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, 2000.

[19] M. Swain and B. Ballard. Color indexing. *International Journal of Computer Vision*, 7, 1991.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, 2001.