The SfinX Video Surveillance System

Raju Rangaswami, Zoran Dimitrijević, Kyle Kakligian, Edward Chang, Yuan-Fang Wang
{raju@cs, zoran@cs, smallart@cs, echang@ece, yfwang@cs}.ucsb.edu
University of California, Santa Barbara

Abstract

In a surveillance system, video signals are generated by multiple cameras with or without spatially and temporally overlapping coverage. These signals need to be compressed, fused, stored, indexed, and then summarized as semantic events to allow efficient and effective querying and mining. This paper presents the hardware and software architecture of SfinX*, a next-generation video-surveillance system. We analyze each component within the software architecture and identify research issues. Finally, we present preliminary results on the performance of various components of* SfinX.

1 Introduction

Video surveillance has been a key component in ensuring security at airports, banks, casinos, and correctional institutions. More recently, government agencies, businesses, and even schools are turning toward video surveillance as a means to increase public security. With the proliferation of inexpensive cameras and the availability of high-speed, broad-band wired/wireless networks, deploying a large number of cameras for security surveillance has become economically and technically feasible. However, several important research questions remain to be addressed before we can rely upon video surveillance as an effective tool for crime prevention, crime resolution, and crime prosection. SfinX (multi-Sensor Fusion and mINing Xystem) aims to develop several core components to process, transmit, and fuse video signals from multiple cameras, to mine unusual activities from the collected trajectories, and to index and store video information for effective viewing [4].

The current state-of-the-art in commercial video surveillance equipment typically consists of analog cameras and tape-based VCRs which are functionally very limited. For instance, these systems do not support simultaneous recording and reviewing of camera data. Analog data on tape must be first converted to digital format before it can be subjected to further analysis. Moreover, retrieval of archived videos is manual and therefore time-consuming. All these issues make current commercial systems obsolete. Current and future surveillance systems must be all digital, capable of handling multiple simultaneous viewing and recording sessions, automatically detect suspicious activity, and most of all, be affordable. To this end, we propose to use cheap off-the-shelf digital video cameras and desktop computers to store, retrieve, analyze, and query the captured videos. Our architecture requires only one high-end camera possessing zoom and motion capabilities for tracking objects or humans in close-up.

The target application that we intend to support would not only be capable of viewing video streams in real-time, but also able to support scan operations (like rew, ffwd, slow-motion, etc.) on the video streams. In addition, it would also support video analysis in the form of database queries. A query, for instance, can be worded like this: "select object = *'vehicles'* where event = *'circling'* and location = *'parking lots'* and time = *'since 9pm last night'*." Another example-query might be "select object = *'vehicle A'* where event = '*' and location = '*' and time = *'since 9pm last night'*."

In this paper, we make the following contributions:

1. We propose the architecture of a next-generation video-surveillance system which not only supports real-time monitoring and storage of all the video streams, but also performs video analysis and answers semantic database queries.

2. We analyze each component of the proposed architecture and present the research problems that need to be solved in order to build a successful video-surveillance system.

3. We present preliminary results of the performance of certain components of the system.

In recent times, there has been a renewed interest in designing all digital video surveillance systems [1, 8, 9, 4, 10, 5]. However, a number of reserch problems remain to be solved before we can build efficient and reliable surveillance systems. We outline the major components within SfinX and associated research problems in Section 3.

2 System Architecture

In this section, we introduce the hardware and software architecture of the SfinX system.

Figure 1 depicts a typical hardware architecture of SfinX. Cameras are mounted at the edges of a sensor network to collect signals (shown on the upper-right of the figure). When activities are detected, signals are compressed and transfered to a server (lower-left of the figure). The server fuses multi-sensor data and constructs spatio-temporal descriptors to de-

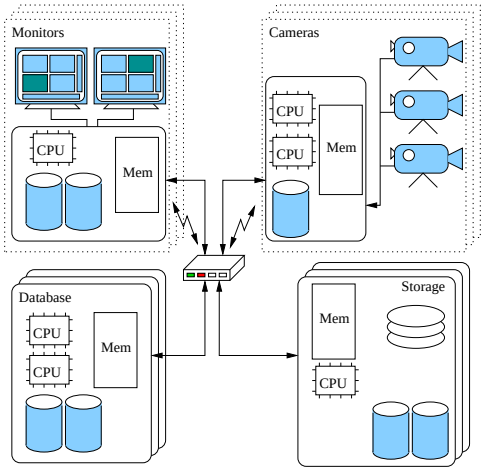

Figure 1. Hardware architecture.

pict the captured activities. The server indexes and stores video signals with their meta-data on RAID storage (lower-right of the figure). Users of the system (upper-left of the figure) are alerted to unusual events and they can perform online queries to retrieve and inspect video-clips of interest.

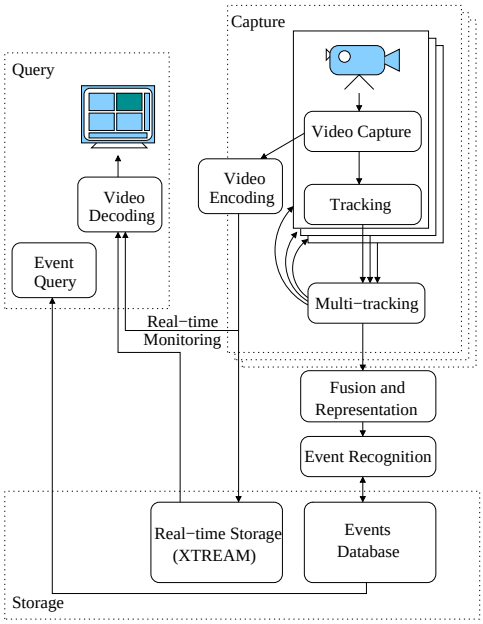

Figure 2. Software architecture.

Figure 2 depicts the software architecture of SfinX. Video signals are captured by the *video capture* module. At the same time *tracking* algorithms are employed to track objects in the captured video streams and the video stream is *encoded* and sent off to be stored onto Xtream [2], a *real-time streaming storage* system. To aid in effective tracking of occluded objects and to obtain consensus on object position in ambiguous situations, a *multi-tracker* module combines the tracking information from different cameras which cover a common physical area and feeds back global information to the individual camera tracking modules. There exist multiple multi-trackers, which track objects in physically disjoint areas.

Using the global tracking information and object representation created by the multi-tracker modules, the *fusion and representation* module maps the trajectory of each object as it moves through the entire scene. The representation module represents the trajectory of each object using *sequence data representation* [9]. This information is stored in the *events database* for future reference.

The user-interface consists of two distinct components. First, the *real-time monitoring* component using which a user can view live camera feeds as well as interact with live feeds to scan through the stream. This helps the user to immediately track objects by moving through the stream at will. Second, the viewer can also analyze the stored video streams by performing database queries. An example of such a query was presented in Section 1. Controlling the query semantics, the user can get detailed information from the database.

3 System Components

In this section, we present the major components of the SfinX system. We analyze each component of the software architecture and describe the interaction between various components.

3.1 Video Capture

For capturing video streams, we propose using multiple, cheap, off-the-shelf video cameras for each physical location requiring surveillance. These cameras share data between themselves to perform their functions with greater accuracy. Similar to a previous study [10], we use a single high-end camera per location possessing zoom and motion capabilities for tracking objects or humans in close-up. The most important problem in capturing useful information from a scene is that of camera callibration [5]. Ideally, this must be an automatic process, that maps the camera co-ordinates to co-ordinates in the physical location. In addition, the close-tracking high-end camera must be perfectly callibrated at all times inspite of zoom and motion operations.

3.2 Encoding and Real-time Storage

The video stream obtained from each camera is encoded using standard encoding algorithms like H.263, MPEG1, or MPEG4. Each stream is then stored using a real-time storage system like Xtream [2] for future viewing purposes. The storage system provides real-time stream retrieval and supports scan operations like rew, ffwd, and slow-motion. The main sub-components of the real-time storage component are: *data placement, admission control, disk scheduling*, and *backup manager*.

The *data placement module* makes decisions about data placements using global knowledge about all storage nodes and the QoS requirements for each IO request. The placement decisions can be short-term (e.g., for each database update) or long-term (e.g., the placement for the next one hour of a particular video stream). The data placement module consults the *admission control* module to check if a particular placement satisfies the real-time access requirements. It also manages data redundancy for reliability.

The *disk scheduling module* is responsible for local disk scheduling and buffer management on each storage node. SfinX uses time cycle scheduling [7] for guaranteed-rate real-time streams. The basic time cycle model is extended to support non real-time IO requests with different priorities (high-priority, best-effort, and background IO). To achieve short latency for high-priority requests while maintaining high disk throughput, SfinX uses preemptible disk scheduling [3].

The *backup manager* module is responsible for deciding which data to copy from main storage to backup and when. The volume of video data in SfinX is large, of the order of TB/day. Since the main SfinX storage is designed to be reliable, backup is mainly used to filter its data and to keep only the important data in the main storage.

3.3 Tracking and Multi-tracking

Tracking refers to the process of following and mapping the trajectory of a moving object in the scene. Moving objects in each camera feed are tracked using real-time tracking algorithms [8, 1]. Using the information about motion trajectory, the high-end camera may be used to follow the moving object in close-up.

Multi-tracking combines the tracking information from different cameras which monitor the same physical location. It uses the global knowledge thus obtained to aid in tracking objects which are occluded for individual cameras. It can also use this global information to reach consensus when individual tracking modules disagree on object positions. The multi-tracker feeds this global information back to the individual camera tracking modules. Each physical location employs a multi-tracker to combine the information from individual cameras in that location.

3.4 Fusion and Representation

Using the global tracking information and object representation created by the multi-tracker modules, the *fusion and representation* module maps the trajectory of each object as it moves through the entire scene. The representation module represents the trajectory of each object using *sequence data representation* [9]. To arrive at a reasonable representation, the trajectory of each object is smoothed using Kalman filters [6] to obtain a piecewise linear trajectory. This piecewise linear trajectory is then represented using sequence data representation.

3.5 Event Recognition

Event recognition translates to the problem of recognizing spatio-temporal patterns under extreme statistical constraints. It deals with mapping motion patterns to semantics (e.g., benign and suspicious events). Recognizing rare events comes up against two mathematical challenges. First, the number of training instances that can be collected for modeling rare events is typically very small. Let N denote the number of training instances, and D the dimensionality of data. Traditional statistical models such as the Hidden Markov Model (HMM) cannot work effectively under the $N < D$ constraint. Furthermore, positive events (i.e., the sought-for hazardous events) are always significantly outnumbered by negative events in the training data. In such an imbalanced set of training data, the class boundary tends to skew toward the minority class and hence results in a high incidence of false negatives.

3.6 Querying and Monitoring

Monitoring allows retrieving videos efficiently via different access paths. Video data can be accessed via a variety of attributes, e.g., by objects, temporal attributes, spatial attributes, pattern similarity, and by any combinations of the above. We support retrieval of videos with trajectories that match a given SQL query definition. At the same time the storage system must also support viewing of stored videos. The infrastructure also supports real-time monitoring of camera streams. However, simultaneously supporting high-throughput writes (recording encoded videos) and quick response reads (retrieving video segments relevant to a query) presents conflicting design requirements for memory management, disk scheduling, and data placement policies at the storage system.

4 Results

In this section, we present results obtained while measuring camera performance, vidoe compression efficiency, network capability, and storage performance.

4.1 Camera Performance

Table 1 presents the performance characteristics of the high-end camera that we currently use to track objects in close-up.

4.2 Compression results

Of the four compression methods we tested (H.263, MPEG4, MSMPEG4, MPEG1), all were within approximatly 8% CPU usage of each other. MSMPEG4 was the slowest, though it did exhibit the best quality. Here we present the compresion results using MPEG1 encoding. Experiments were carried out on an Intel P4 2.66GHz using an open source video encoder, *ffmpeg*.

We notice an interesting trend in Figure 3, which depicts the CPU utilization to compress video MPEG1 at 15 fps at

Parameter	Value
Camera model	Sony EVI-D30
Output resolution	460x350 NTSC tv lines
Pan range	193.75 degrees (specs say 200 degrees)
Maximum pan speed	80.7 degrees/sec (specs say 80)
Pan accuracy	± 0.45 degrees approx.
Tilt range	47.36 degrees
Maximum tilt speed	52.6 degrees/sec (specs say 50)
Tilt accuracy	± 0.22 degrees approx.
Zoom ("tele" and "wide")	1x to 12x at 6 speed settings

Table 1. Measured performance parameters for the Sony EVI-D30 high-end camera.

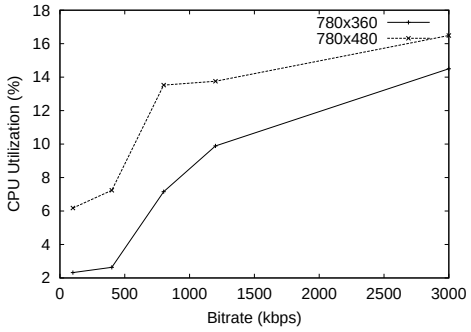

Figure 3. CPU utilization for compression.

different resolutions. A larger resolution video naturally requires more CPU. Also, the larger the bitrate, the larger the CPU usage. This is counter intuitive because one would think the more you compress the video, the more work it requires. In addition to this chart, we found that capturing at 15fps uses a little over half of the CPU as capturing at 30fps.

4.3 Network streaming results

Bit-rate (kbps)	# Streams
100	919
400	229
800	114
1200	76
3000	30

Table 2. Throughput of 100 Mbps network.

Table 2 gives an estimate about the number of streams that can be supported at various bit-rates over a 100Mbps local area network. The network is switched ethernet and the switch is a HP 2324 Procurve box.

4.4 Storage results

We now present results for real-time storage using Xtream [2]. We use an Intel Pentium 4 1.5 GHz Linux based PC, with 512 MB of main memory and a WD400BB 40 GB hard drive. The maximum sequential disk throughput is 31 MBps in the fastest zone and 21 MBps in the slowest zone.

We performed experiments for the following two scenarios: homogeneous constant bit-rate (type C) and variable (type V) bit-rate streams where all serviced streams have the same bit-rate. N denotes the maximum number of streams that the system can support without missing deadlines.

Avg. BR	N: Type C	N: Type V
250 kBps	44	44
1000 kBps	20	23
2000 kBps	12	11

Table 3. Disk throughput.

5 Conclusion

In this paper, we have described the architecture and design of SfinX, a next-generation video-surveillance system. We have enumerated the research challenges and requirements for each component of the system and outlined our solutions. Although SfinX is oriented to support surveillance applications, its components will continue to involve research in the areas of Computer Vision, Signal Processing, Machine Learning, Databases, and Systems.

References

[1] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa. A system for video surveillance and monitoring. *Robotics Institute, Carnegie Mellon University Technical Report*, (CMU-RI-TR-00-12), May 2000.

[2] Z. Dimitrijevic, R. Rangaswami, and E. Chang. The XTREAM multimedia system. *Proceedings of the IEEE Conference on Multimedia and Expo*, August 2002.

[3] Z. Dimitrijevic, R. Rangaswami, and E. Chang. Design and implementation of Semi-preemptible IO. *Proceeding of the 2nd Usenix FAST*, March 2003.

[4] Z. Dimitrijevic, G. Wu, and E. Chang. SFINX: A multi-sensor fusion and mining system. *Proceedings of the IEEE Pacific-rim Conference on Multimedia*, December 2003.

[5] T. Gandhi and M. Trivedi. Motion analysis of omni-directional video streams for a mobile sentry. *ACM International Workshop on Video Surveillance*, November 2003.

[6] E. Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[7] P. V. Rangan, H. M. Vin, and S. Ramanathan. Designing and on-demand multimedia service. *IEEE Communications Magazine*, 30(7):56–65, July 1992.

[8] C. Stauffer and E. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[9] G. Wu, Y. Wu, L. Jiao, Y.-F. Wang, and E. Chang. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. *Proceedings of the 11th Annual ACM International Conference on Multimedia (ACMMM)*, 2003.

[10] X. Zhou, R. Collins, T. Kanade, and P. Metes. A master-slave system to acquire biometric imagery of humans at distance. *ACM International Workshop on Video Surveillance*, November 2003.