Content Description Servers for Networked Video Surveillance

Jeffrey E. Boyd Maxwell Sayles Luke Olsen
Paul Tarjan
Department of Computer Science, University of Calgary
boyd@cpsc.ucalgary.ca

Abstract

Advances in digital signal processing technology make it possible to embed the low-level functions performed by a video surveillance system into cameras. The result is metadata cameras, like MPEG-7 cameras, that provide descriptions of what they see in the form of XML documents served over a network. We use this concept to build video information servers that are conceptually similar to MPEG-7 cameras, but differ in that they interact with client applications and can be configured dynamically to perform different functions. We demonstrate the use of these video information servers in some simple surveillance applications. Dynamic configuration of the servers allows them to do more than describe video content. For example, we present a functioning surveillance system, built with content description servers, that describes the activity in a hockey game based on the observations of several cameras. Thus, we demonstrate that content description servers provide building blocks with which to create networked video surveillance systems.

Keywords: networked surveillance, XML, MPEG-7, content description

1. Introduction

Machine-vision based video surveillance systems perform several common functions required for low-level processing of video data. These algorithms include:

- acquisition of images from a camera,
- calibration of a camera [8],
- segmentation of an image into foreground and background pixels [19, 9, 18, 12],
- location of objects that are connected components of foreground pixels,
- computation of object descriptions, and
- object tracking within a camera's field of view [7, 16, 5, 1].

In addition to these low-level functions, surveillance systems may also perform

- object tracking in world coordinates,
- behavior recognition from object trajectories, and
- storage and retrieval of trajectories for long-term analysis,

among others.

Recent advances in the speed and size of digital signal processing hardware make it possible to embed many of the low-level functions listed above into a video camera. For example, see the work of Wolf et al. [20, 14]. Cameras with such embedded functionality have the option to provide not only raw pixels, but metadata that describe the locations and motion of objects viewed by the camera. The availability of metadata simplifies the implementation of surveillance systems that perform the high-level functions listed.

While cameras with embedded metadata production can transmit data over a variety of media, use of computer networks is convenient because it uses established infrastructure as well as exploits the abundance of network devices that are already available. In order to distinguish metadata cameras from network cameras that provide only compressed video, we refer to these cameras as *video information servers*. Sections 2 and 3 of this paper describe the video information servers that we have built for surveillance. We also describe surveillance systems built with these servers.

Video information servers are an example of a broader class of *content description servers*, systems that process data gleaned from either sensors or other servers to produce a description of a scene that they, in turn, serve to other systems. In Section 5, we describe one such system that observes a hockey game using several video information servers, builds a model of the game, and serves data from that model to client applications.

2. Content Description Camera

2.1. Camera Markup Language Servers

Boyd et al. [3] describe a video surveillance system that assimilates information extracted from multiple cameras into a single model of a scene. In a real-time implementation, separate processors perform the low-level operations (motion segmentation and object detection), and then share, via a local-area network, the information with other parts of the surveillance system. Sayles et al. [17] describe their *CaML* (*Camera Markup Language*) network camera system, that extracts the low-level processing portions of the Boyd et al. [3] system to build a video information server. Their system provides, via network connections, a description of camera functions, a raw video stream, and a stream of XML documents that describe detected objects and their trajectories in image coordinates. To detect moving objects, the CaML system computes Lucas-Kanade optical flow [13], thresholds the flow, and identifies large connected components of *moving* pixels. The system is implemented in the Python programming language, and makes use of the Intel Integrated Performance Primitives Library [10] to optimize execution speed. Sayles et al. demonstrate their CaML server system with a client application that combines the image-space trajectories and camera calibration data (provided by the servers) to compute trajectories in a three-dimensional model of an area under surveillance.

2.2. Improvements

The work described in this paper uses the video servers described by Sayles et al [17] with the following variations.

- The servers provide separate object and trajectory descriptions. Whereas the set of objects consists of all moving objects detected by the server, the trajectories are formed when the server solves the data association problem to connect objects detected in a series of frames. This distinction allows client applications to use or ignore the data association performed by the server as needed.
- Communication with the server is bidirectional. Clients can change the behavior of the server by sending documents to it (see Section 3).
- To deal with conflicting requests from multiple clients, client applications can include a user name and password in documents that make requests to the server. The server maintains a list of privileges associated with each user. Thus, the server can restrict pan-tilt-zoom control to only a few clients to prevent conflicts in aiming the camera, and at the same time provide object descriptions to all clients.
- The servers are dynamically configurable. Privileged clients can change the algorithms that the server uses for its processing, and even the tags that the server uses in the documents that it provides.
- Camera calibration is simplified. Whereas previous servers provided a complete three-dimensional calibration in the form of a camera matrix, they now provide a projective transformation that relates image coordinates for different pan-tilt-zoom settings. This modification is inadequate for three-dimensional tracking, but is simpler to implement, allows warping of all images from a stationary pan-tilt-zoom camera into a common reference frame, and is sufficient for tracking objects constrained to move on a two-dimensional surface [8].

At the time of writing, we have a CaML video information server running continuously, observing traffic at the University of Calgary. A Java applet at http://toaster.cpsc.ucalgary.ca/ shows the data available from this server. (*For ITCC reviewers, user = vision, password = lab. Note that while this system operates continuously, it requires daylight to detect objects reliably. It is situated in the mountain time zone.*)

2.3. MPEG-7 Cameras

CaML video information servers are similar to recent MPEG-7 cameras (e.g., see Ebrahimi et al. [6]) in many respects. Both combine raw video with metadata in the form of XML documents. However, the video information servers described in this paper differ from an MPEG-7 camera in the following ways.

- Although the creators of MPEG-7 anticipate a wide array of applications [15], we feel that MPEG-7 is biased toward storage and retrieval applications, and relies on the extensibility of XML for other applications. We use our own set of XML tags suited to our application.
- MPEG-7 describes a uni-directional flow of content description, whereas the video servers allow interaction through the exchange of documents.
- Dynamic reconfiguration of the server allows clients to change the behavior of the server.

3. Dynamic Configuration

While the functionality and low-level video processing requirements of the video information servers motivated

```
<CaML></CaML>
```

(a)

```
<CaML>
    <camera load="ControllerModule">
    </camera>
</CaML>
```

(b)

```
<CaML received_at="12.926206"
        generated_at="12.927514"
        version="2.0c" id="breadmaker">
    <camera loaded="ControllerModule">
        <min_pan>-880</min_pan>
        <max_pan>880</max_pan>
        <min_tilt>-300</min_tilt>
        <max_tilt>300</max_tilt>
        <min_zoom>0</min_zoom>
        <max_zoom>1023</max_zoom>
        <pan>0</pan>
        <tilt>0</tilt>
        <zoom>0</zoom>
        <camera_matrix>
            1.000000 0.000000 0.000000
            0.000000 1.000000 0.000000
            0.000000 0.000000 1.000000
        </camera_matrix>
    </camera>
</CaML>
```

(c)

```
<CaML>
    <camera>
        <pan>150</pan>
        <tilt>250</tilt>
        <zoom>0</zoom>
    </camera>
</CaML>
```

(d)

Figure 1. Sample CaML documents: (a) a minimal CaML document, (b) a document that request the server to load a camera control module, (c) the server response to the document in (b), and (d) a document that a client can send to request changes in pan, tilt, and zoom settings.

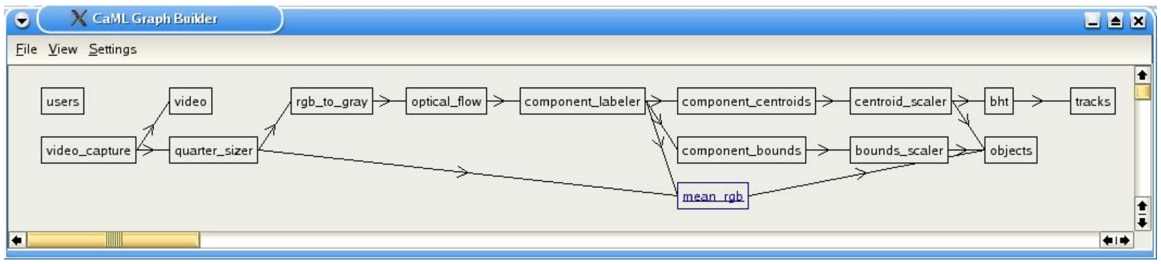


Figure 2. Screen capture of CaML graphical configuration editor. The editor creates CaML documents that can be used to configure a CaML server for a desired task. Note that the editor can send the documents to the server as they are edited, allowing a user to tune server performance interactively.

their design, we can view them as generic systems that process and exchange information by exchanging XML documents over a network. Video information servers perform the following salient functions:

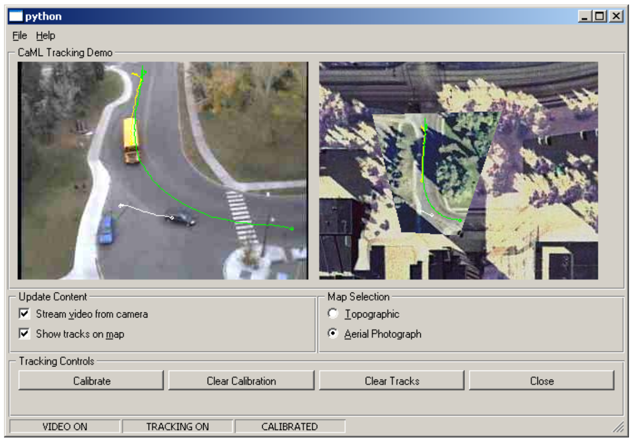
- acquire information from an external source,
- process the information as requested by clients, and
- serve the resulting information to clients in XML documents.

Our video information servers are implemented as generic content descriptions servers. The server takes on the role of video information server only when configured to do so by requests from client applications. Clients make these requests by sending documents to the server directing the server to load modules that provide the desired functionality.

For example, Figure 1(a) shows a minimal CaML document. Before the server can recognize sub-tabs, it must load a module. Figure 1(b) shows a document that directs the server to load a module to control basic camera functions such as pan, tilt, and zoom. The server responds with the document in Figure 1(c). Subsequently, the client can configure camera settings by sending a document, like that shown Figure 1(d), to the server.

It is also possible to terminate a module by unloading it. Thus, by utilizing this capability, we can *dynamically reconfigure* generic content description servers to perform a variety of roles. For video information servers, we have modules to perform tasks such as user access control, video capture, video compression, object detection (one module based on optical flow and another based on color segmentation), and object tracking. Dynamically loadable modules provide a mechanism to update a server when improved algorithms become available. Note that by changing the module responsible for generating XML tags for object description, the server could switch from serving CaML documents to MPEG-7 documents.

Modules operate in a pipeline architecture where the output of a module can provide input to one or more other modules in a pipeline. The modules are task-specific and cannot be connected arbitrarily. Therefore, configuration of the modules and the pipeline requires knowledge of the module's functionality. To facilitate server configuration, we created a graphical pipeline editor. The editor allows a user to graphically specify the topology of the pipeline and configure each module. When the user is satisfied with the configuration, the editor creates a CaML document that, when sent to a CaML server, will cause the server to load and configure the desired modules. Note that the editor can send the documents directly to a server, allowing a user to view immediately the effects of altering module parameters. Figure 2 shows the graphical configuration editor in use.



[✓] Stream video from camera
[✓] Show tracks on map
Topographic [✓ Aerial Photograph]

VIDEO ON | TRACKING ON | CALIBRATED

Figure 3. Sample video surveillance client application. The application shows live video on the left with object trajectories superimposed. A manual calibration allows the application to transform and superimpose object trajectories on the aerial photograph on the right. The application inverts pixels in the aerial photograph, outside the camera's field of view.

4. Surveillance Applications

We demonstrate the operation of video information servers with two client applications.

Figure 3 shows the first client application in operation. The image on the left is live video provided by the video information server. The client superimposes plots of moving object positions and trajectories on the raw video. The image on the right shows an aerial view of the scene under surveillance. A manual calibration, done with the client application, finds a projective transformation that transforms coordinates in the camera image to coordinates in the aerial view. This allows the client to plot object trajectories superimposed on the aerial view. Alternatively, the application can use a map of the surveillance area in place of the aerial photograph. This allows automated interpretation of the scene, in a natural coordinate system. For example, if we wish to scrutinize traffic in a particular region, we can define that region in map coordinates, independent of the camera's position. Furthermore, the camera calibration data provided by the video information server will allow the defined region to remain fixed on the ground, even as the camera pans, tilts, and zooms.

Figure 4 shows a client application that observes traffic to collect statistics to estimate long-term traffic patterns using the network tomography [2]. The image on the left

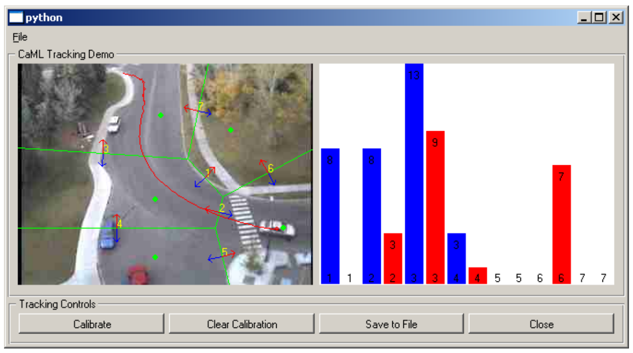


Figure 4. Sample video surveillance application. The application shows live video on the left with object trajectories superimposed. A user segments the scene into several regions using a Voronoi diagram constructed interactively. The application then keeps track of the number of vehicles that cross the defined boundaries. Data collected in this manner can be used to estimate long-term traffic patterns while minimizing biased, systematic errors due to dropped tracks, using the method of Boyd and Meloche [2].

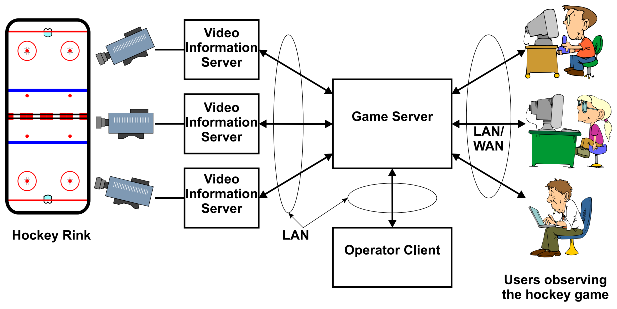
shows the raw video stream with object trajectories superimposed. A user defines regions of interest in the image using a graphical editor to manipulate a Voronoi diagram. The Voronoi diagram establishes the boundaries between the regions. The application detects vehicles as they cross the boundaries. It then adds these crossing events to its statistical tally, shown on the right.

Note that for both of these client applications, the client does little or no processing of the raw video data. In fact, each client can operate with as little as one frame of video data (either to calibrate the first client, or to delineate regions in the second), using only the object trajectory descriptions thereafter.

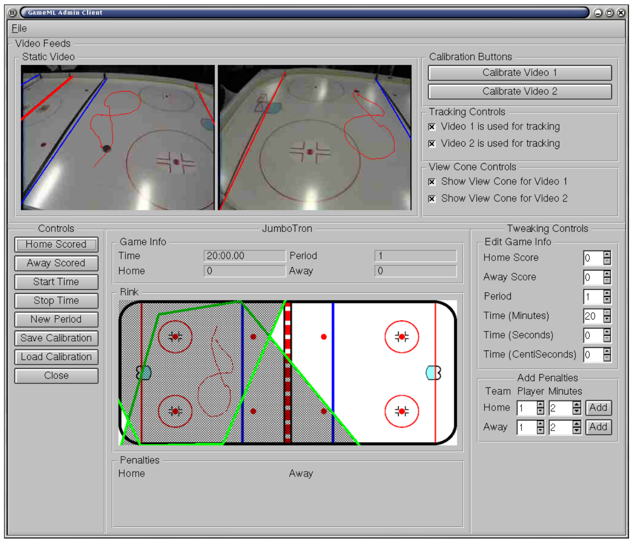
5. Environment Model Systems

Content description systems can do more than serve descriptions of images from a camera. For example, Figure 5(a) shows a system that performs surveillance of a hockey game and serves descriptions of the game to viewers for entertainment purposes. The system has two types of servers: video information servers (one associated with each camera observing the game) and one game state server. Figure 5(b) shows the operator client in use. The client plots the trajectories in both camera and rink coordinate systems. Shaded regions superimposed on the map of the rink show the area covered by each of two cameras.

Upon start up, the game server is identical to the camera servers. Documents sent by the operator client configure the server to perform its role, i.e., to form an *environment model* (Jain and Wakimoto [11]). Thus, dynamically reconfigurable content description servers provide building blocks with which to create surveillance systems. Configured as video information servers, they perform low-level processing on video sources, but they can also be configured to assimilate the information derived from multiple sources and to serve that information to other systems. Boyd et al. [4] describes the operation of the hockey game surveillance system in detail.



(a)



(b)

Figure 5. A surveillance system that serves descriptions of actions in a hockey game: (a) the system architecture, and (b) the operator client in use showing information provided by the game server. Both the camera and game servers are dynamically configured content description servers.

6. Conclusions

Dynamically configurable content description servers provide an architecture upon which to build network surveillance systems. In a basic application, a content description server can be a metadata camera, such as those that provide MPEG-7 content description. However, interaction and dynamic configuration allow these servers to process and serve high-level information in a surveillance system. These server systems can perform basic surveillance functions, thus facilitating the creation of more complex surveillance systems.

References

[1] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Academic Press, Boston, Massachusetts, 1988.

[2] J. Boyd and J. Meloche. Evaluation of statistical and multiple-hypothesis tracking for video traffic surveillance. *Machine Vision and Applications*, 13(5-6):344–351, 2003.

[3] J. E. Boyd, E. Hunter, P. Kelly, L. Tai, C. Phillips, and R. Jain. Mpi-video infrastructure for dynamic environments. In *IEEE Conference on Multimedia Systems 98*, Austin, TX, June 1998.

[4] J. E. Boyd, M. Sayles, Luke Olsen, and Paul Tarjan. Internet broadcast of hockey: a scale prototype. In *SPIE Internet Imaging V*, Santa Jose, CA, January 2004. accepted.

[5] I. J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[6] T. Ebrahimi, Y. Abdeljaoued, R. Figueras i Ventura, and O. Divorra Escoda. Mpeg-7 camera. In *ICIP01*, pages Multimedia Indexing, Browsing, and Retrieval, 2001.

[7] A. Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, Massachusetts, 1974.

[8] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, Cambridge, UK, 2000.

[9] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real time robust background subtraction. In *IEEE Frame Rate Workshop*, 1999.

[10] Intel Corporation. *Intel Integrated Performance Primitives for Intel Architecture: Reference Manual*, 2003.

[11] R. Jain and K. Wakimoto. Multiple perspective interactive video. In *IEEE International Conference on Multimedia Computing and Systems*, pages 202–211, May 1995.

[12] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *IEEE Workshop on Motion and Video Computing*, pages 22–27, Orlando, FL, December 2002.

[13] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[14] T. Lv, B. Ozer, and W. Wolf. Smart camera system design. In *International Packet Video Workshop 2002*, Pittsburgh, PA, April 2002.

[15] B. S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7*. John Wiley and Sons, Chichester, England, 2002.

[16] D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 6(AC-24):843–854, 1979.

[17] M. Sayles, X. Wu, and J. E. Boyd. Caml: Camera markup language for network interaction. In *SPIE Internet Imaging IV*, volume 5018, pages 248–256, Santa Clara, CA, January 2003.

[18] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition 99*, volume II, pages 246–252, 1999.

[19] E. Sudderth, E. Hunter, K. Kreutz-Delgado, P. H. Kelly, and R. Jain. Adaptive video segmentation: theory and real-time implementation. In *1998 Image Understanding Workshop*, volume 1, pages 177–181, November 1998.

[20] W. Wolf, B. Ozer, and T. Lv. Smart cameras as embedded systems. *Computer*, 35(9):48–53, September 2002.