Internet Broadcast of Hockey: A Scale Prototype

Jeffrey E. Boyd, Maxwell Sayles, Luke Olsen, Paul Tarjan

Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4

ABSTRACT

We present a system for the broadcast of hockey games over the internet. The system allows users to experience the hockey game while it is in progress. Our system uses generic *content description servers* that acquire information from an external source, process it, and serve the processed data to client systems. Dynamic configuration of the servers allows us to use them in a variety of roles. For example, *video information servers*, like an MPEG-7 camera, produce XML documents that describe the motion of objects in the scene in addition to unprocessed video. Unlike an MPEG-7 camera, our video information servers interact with client systems, and can change their behavior through dynamic configuration. In an alternate configuration, a content description server acts as a *game server* in our hockey broadcast system. The game server forms an environment model that encapsulates the state of the hockey game and serves data from the model to clients. We developed and tested our system using a 1/32-scale model of a hockey rink. Early results using data acquired at a real rink indicate that the system performs as expected.

Keywords: sports, internet, broadcast, trajectory, camera, XML, motion, video, MPEG-7

1. INTRODUCTION

Jain[6] suggests that the future of multimedia systems lies in a user's experience of the events that a system portrays. We maintain that such an experience can only be obtained by providing live coverage of an event. While archival and retrieval of data are useful, to truly experience events such as sports requires that a multimedia system deliver data in real-time, as the event occurs. The difference may be likened to that of viewing a sporting event live, versus watching a recording. The only way to truly enjoy the recording is to have absolutely no knowledge of the outcome, and without use of a *fast forward* to speed through lulls in play.

In this paper we present a system that extracts data from multiple sources to describe the state of a hockey game in real-time. Our system is loosely based on the concept of multiple-perspective interactive video (MPI-Video) described by Jain and Wakimoto.[7] In this sense, we build an *environment model* that contains a description of the hockey game. Our system does this as the game occurs, delivering game descriptions over a network to allow viewers to experience the event.

To build our system, we developed novel *content description servers*. These servers acquire information from an external source, process the information and serve it to client applications. For example, a *video information server* is a content description server that acquires data from a video camera, extracts information about moving objects in a scene, and serves that information in the form of XML documents. We also demonstrate a hockey *game server* that acquires information from a score board and multiple video information servers, forms a model of the game state, and serves information from that model.

Clients of content description servers select the data they receive and how they utilize it. For example, a complex client with a high-bandwidth connection to our game server can selectively view any video stream from video servers, plot player trajectories, and obtain score board information. Alternatively, a client that has only a low-bandwidth connection can choose not to acquire the video, and display the game in an abstract manner, showing game activity as a set of trajectories plotted on a map of the rink.

We demonstrate our system using a 1/32-scale model of a rink. We also show early results from a real rink using recorded data from a single camera.

Send correspondence to: boyd@cpsc.ucalgary.ca.

2. CONTENT DESCRIPTION SERVERS

Our internet hockey broadcast system is an interconnected network of servers and clients. In this section we describe the operation of the servers beginning with the cameras, i.e., *video information servers*, and extending this to the more general *content description servers*.

2.1. Video Information Servers

Boyd et al.[1] describe an MPI-Video system that assimilates information extracted from multiple cameras into a single environment model. Their environment model consists of raw images, segmented images, camera calibrations, three-dimensional trajectories, and behavior descriptions. Although described as a layered model with an information hierarchy that ranged from low-level pixels to high-level behavior descriptions, the system functioned as a pipeline, feeding data from low to high-levels in the model. In a real-time implementation of the system, separate processors performed the low-level operations such as motion segmentation and object detection, then added the information to the environment model via a local-area network.

Sayles et al.[10] describe their *Camera Markup Language* (*CaML*) network camera system, that extracts the low-level processing portions of the Boyd et al.[1] 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. They demonstrate their CaML video information servers 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.

The work described in this paper uses the video servers described by Sayles et al[10] with the following variations and improvements.

- The servers provide separate object and trajectory descriptions. Whereas the set of objects includes 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.
- To avoid conflicts between clients, the server distinguishes between ordinary clients and privileged clients. A client is privileged when it provides the correct password to the server in a document. For example, the server can restrict pan-tilt-zoom control to privileged clients and prevent conflicts in aiming the camera.
- The servers are dynamically configurable. 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 quasi calibration in the form of a projective transformation that maps image coordinates for different pan-tilt-zoom settings to a common coordinate system. This modification is not sufficient for three-dimensional tracking, but is simpler to implement, allows warping of all images from a stationary pan-tilt-zoom camera into a panoramic image mosaic, and is sufficient for tracking objects constrained to move on a two-dimensional surface.[5]

CaML *video information servers* are similar to recent MPEG-7 cameras (e.g., see Ebrahimi et al.[3]) in many respects. They 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 anticipated a wide array of applications ,[8] 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 applications.

```
<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.

- 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.

2.2. Generalized Content Description Servers

While the functionality and low-level video processing requirements of the video information servers motivate their design, we can view them as generic systems that process and exchange information by exchanging documents over a network. Both the video information servers and the game server share the following properties:

- both acquire information from an external source,
- both process the information as requested by clients, and
- both serve the resulting information to their clients.

When the information that these systems serve describes the content of a media stream, then we call these systems *content description servers*.

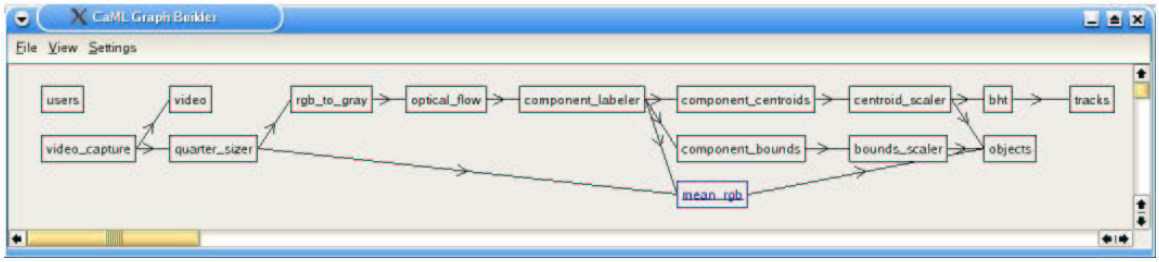


Figure 2. Screen shot of CaML graphical configuration editor. The editor creates CaML documents that can be used to configure a CaML server for a desired task.

In our hockey broadcast system, we implement both the camera/video information servers and the game servers as content description servers. In fact, before interaction begins, both types of servers behave identically. Their behaviors diverge when client requests dictate that they perform different tasks. Clients make these requests by sending documents to the server requesting that the server load modules to provide the desired functionality.

For example, Figure 1(a) shows a minimal CaML document. The server must load a module to enable it to recognize sub-tags. 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. The servers can also *unload* modules to terminate their execution.

By utilizing this capability, we can dynamically reconfigure the 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. 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. Dynamic configuration also provides a convenient mechanism for developing, testing, and upgrading new content-extraction algorithms.

Loaded modules operate in a pipeline architecture where the output of a module can provide input to one or more other modules. The modules are task-specific and cannot be connected arbitrarily. Therefore, configuration of the modules and the pipeline requires knowledge of the modules' 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 load and configure the modules desired. Figure 2 shows a screen shot of the graphical configuration editor. The editor can also send configuration documents to the server while editing to allow interactive tuning of the server.

3. SYSTEM ARCHITECTURE

Figure 3 shows the architecture of our internet hockey broadcast system. A series of static cameras placed around a rink observes the action in a hockey game. Each camera feeds raw analog video to a video information server that identifies objects on the rink. The server then sends compressed video and XML documents to client systems.

The game server acquires data from all of the video information servers. It then forms an environment model[7] that encapsulates the activity in the hockey game by assimilating information acquired from the video servers and the game score board. The game server then serves data from the environment model to client systems.

Users wishing to experience the hockey game do so via client applications that acquire and display data from the game server. A special client, the operator client, provides a mechanism for manual control of the game

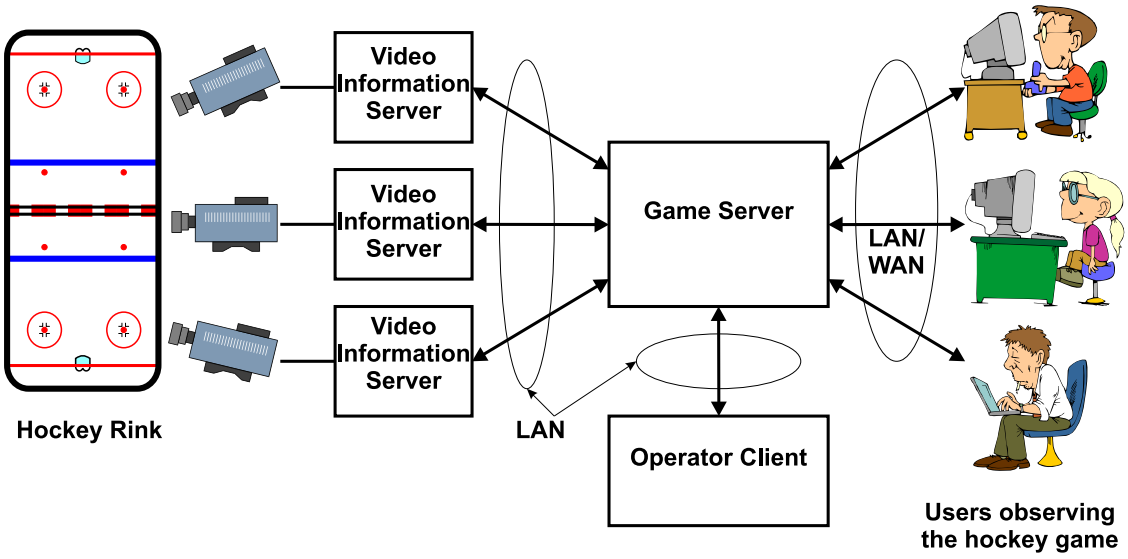


Figure 3. Architecture of our MPI-Video system to facilitate user experience of a hockey game.

server operation. For example, when the game server requires human input to perform a camera calibration, the operator client provides a graphical user interface (GUI) to facilitate the calibration process.

4. ASSIMILATION IN THE GAME SERVER

4.1. The Environment Model

The game server shown in Figure 3 assimilates data from multiple sources into an environment model that represents the game state. The game server acquires data from several sources, include the following.

- Game state information that is typically found on a score board including:
  - the score,
  - the period,
  - time remaining in the period,
  - penalties being served, and
  - time remaining in those penalties.
- A map of the rink.
- Data from video information servers, including
  - images from video stream,
  - camera calibration, and
  - object locations and trajectories.

The environment model formed by the game server contains the following data.

- Score board game state information.
- Raw video.
- Trajectories of players in rink coordinates.

4.2. Score Board Data

Ideally, the game server acquires score-board data directly from equipment operated by game officials. However, in our prototype system, the operator client provides the score-board data. When the clock is running, the game server will automatically run the game clock, but because we cannot precisely synchronize the official game time with the time in the server, the operator GUI allows for fine adjustments to the clock during stops in play. All other score-board information is entered manually with the operator client, and sent to the server in a document.

4.3. Camera Calibration

The environment model keeps all player trajectories in two-dimensional rink coordinates, and therefore the game server must convert image to rink coordinates. To do this we perform a quasi camera calibration by finding a projective transformation, to map screen coordinates to rink coordinates. A human operator manually identifies corresponding points in images and the rink map. Fortunately, a hockey rink has many easily-identified features such as face-off spots and lines. Once the human operator has provided sufficient correspondences (a minimum of four are required, but more is preferable), the operator client computes the projective transformation using methods described by Hartley and Zisserman.[5] The operator client then configures the game server by sending a document containing the transformation to the server.

The operator client GUI facilitates the calibration by plotting the calibrated positions of salient rink markings such as lines and face-off spots over the images from the camera. This allows the human operator to verify visually that the calibration is correct.

Note that this quasi calibration works well with the calibration provided by the video information servers. We need only perform the image-to-rink calibration for one position for a pan-tilt-zoom camera and the game server can derive image-to-rink calibrations for all pan-tilt-zoom settings.

4.4. Tracking

The game server tracks the movement of individual players using a *best-hypothesis tracker* (BHT). Whereas the well-known *multiple-hypothesis tracker* (MHT)[2,9] tracks multiple objects, in clutter, by forming multiple hypotheses for the association of data to tracks, our BHT is simplified to use only the best hypothesis, but can update tracks from multiple data sources. In both the BHT and MHT, once data is associated with a track, the fundamental tracking process is a Kalman filter.[4]

4.5. Clients

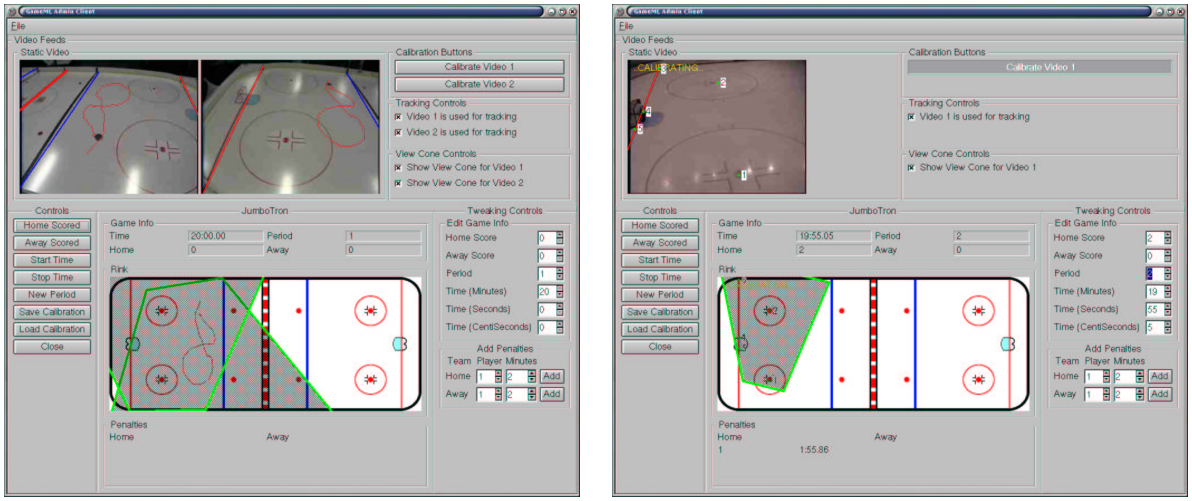
We constructed two client applications that interact with the game server. The first is the operator client (described above). The operator requires high-bandwidth access to the server and presents all the data provided by the game server to the user. Figure 4 shows the operator client in use.

Figure 5 shows a second client designed for use with low-bandwidth connections. This client presents an abstract view of the game, showing only the score-board information and player trajectories mapped onto an image of the rink.

5. RESULTS

Although some hockey rinks have facilities for broadcast cameras, most do not have facilities for installing multiple cameras for experimental multimedia systems. Furthermore, rinks are expensive facilities that are constantly in use in order to justify their cost. It is not a simple matter to obtain access to a rink to install cameras because it would require interruptions in rink usage. For these reasons, we chose to develop our system with a scale model. The scale model served two significant purposes.

1. The scale model allowed us to develop and test the system without requiring access to a real rink. Sufficient realism in the model allowed us to verify our system in a realistic manner, but with the convenience of working in a laboratory.



(a) (b)

Figure 4. Screen shots of the operator client for the game server: (a) game server using two cameras to view scale model, and (b) one camera viewing a real hockey rink, calibration in progress.

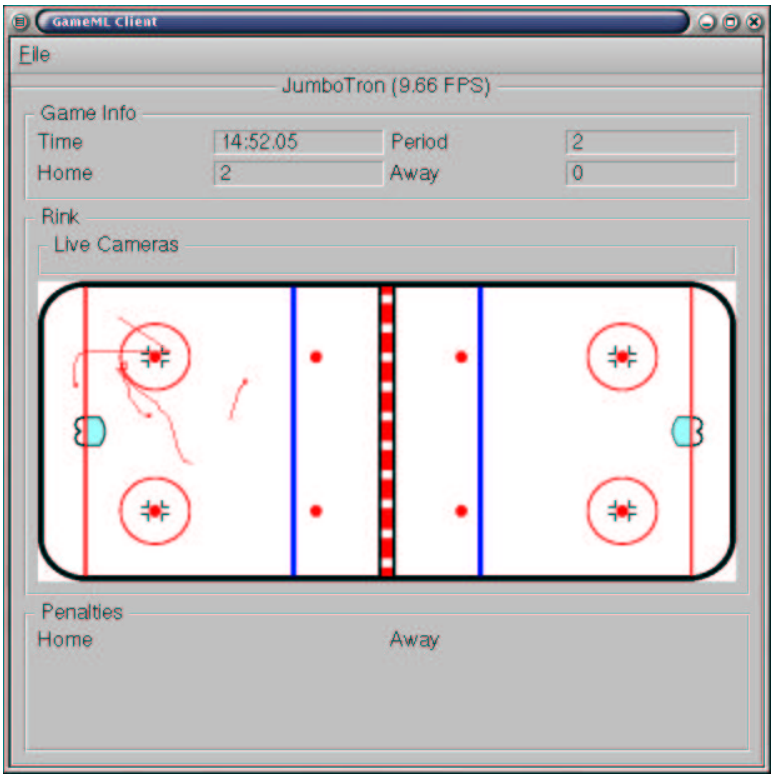


Figure 5. Screen shots of the low-bandwidth client for the game server. Only score board and player trajectory information are displayed.

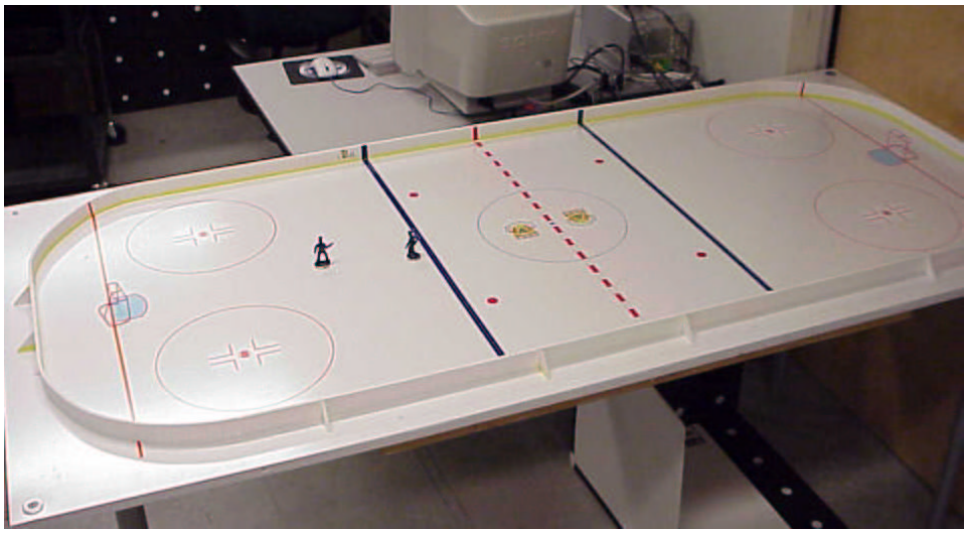
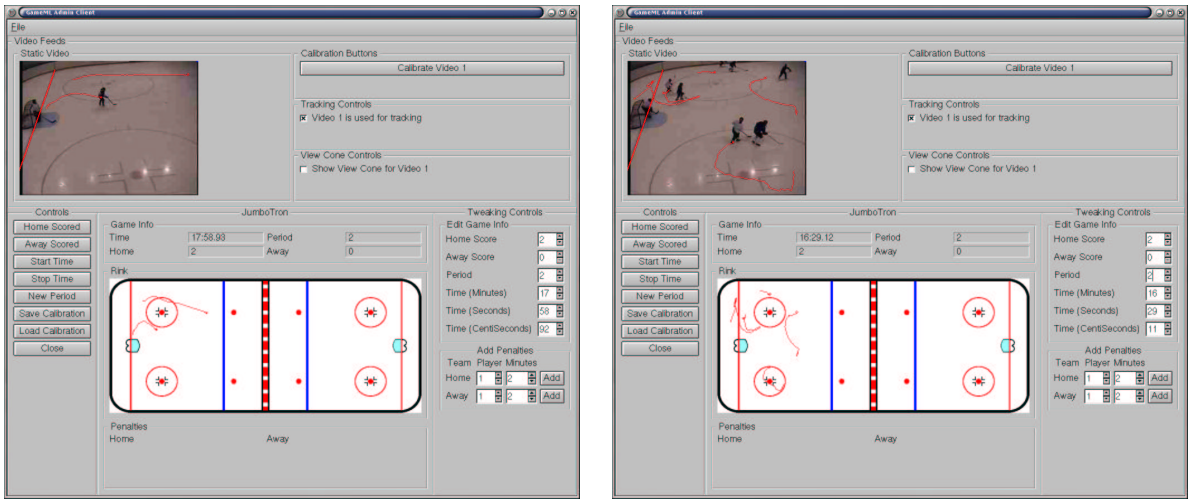


Figure 6. Our 1/32-scale model of a National Hockey League sized rink. Figure 4(a) shows trajectories acquired from the model.



(a) (b)

Figure 7. Screen shots of the operator client for the game server applied to full-size hockey rink.

2. A successful demonstration of our system, with a scale model, provides the kind of demonstration necessary to convince rink operators to allow us to install cameras. In fact, our scale prototype has accomplished this goal.

Figure 6 shows our 1/32-scale model. Although rinks vary in size, our model is based on the 200 by 85 foot National Hockey League norm. All lines, face-off spots, face-off circles, and goal creases are correct to scale. The surface is a matte Plexiglas that simulates the specular reflections produced by a recently resurfaced ice rink. Note that with a real rink, the optical properties of the ice vary through the course of a game as skates gouge the surface. In order to simulate the movement of players on the ice, we used two-inch plastic military figurines mounted on magnets. We moved the figurines by manipulating magnets on the underside of the model.

After initial success with our scale model, we were able to test the system at a real hockey rink. We recorded a hockey practice at the Calgary Olympic Oval (a facility used by the 1988 Winter Olympic Games, with two hockey rinks and a speed skating oval) with a single camera. In the laboratory we played the recorded video into a video information server and the game server. Figure 4(b) shows the calibration in progress for the recorded data. Figure 7 shows a sample of our results. The map of the hockey rink shows the player trajectories superimposed on the video stream and on the rink map. We can see the system tracking and displaying the trajectories of moving players. The system only tracks moving objects, so we often do not see a trajectory for the goalie. When players are in close proximity, the system does not resolve them and instead shows a single track.

6. CONCLUSION

We describe a system for broadcasting hockey over an internet so that viewers can experience the game while it is in progress. The system allows the viewer to observe the game at different levels of abstraction depending on the available bandwidth.

The fundamental building blocks of our system are generic content description servers the produce XML documents that describe a scene, in real-time. These servers interact with client systems and can be configured dynamically to alter their behavior, or even their basic functionality. Examples of our content description servers include video information servers that behave like an interactive, tunable MPEG-7 cameras, and a game server that provides information about the state of a hockey game.

We developed the system using a 1/32-scale model rink to test our ideas. Early results indicate that the system also works well at full-size rinks with real hockey players.

REFERENCES

1. 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.
2. 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.
3. T. Ebrahimi, Y. Abdeljaoued, R. Figueras i Ventura, and O. Divorra Escoda. Mpeg-7 camera. In *ICIP01*, pages Multimedia Indexing, Browsing, and Retrieval, 2001.
4. A. Gelb, editor. *Applied Optimal Estimation.* MIT Press, Cambridge, Massachusetts, 1974.
5. R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge University Press, Cambridge, UK, 2000.
6. R. Jain. Experiential computing. *Communications of the ACM*, 46(7):48–55, July 2003.
7. R. Jain and K. Wakimoto. Multiple perspective interactive video. In *IEEE International Conference on Multimedia Computing and Systems*, pages 202–211, May 1995.
8. B. S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7.* John Wiley and Sons, Chichester, England, 2002.
9. D. B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 6(AC-24):843–854, 1979.
10. 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.