Intelligent Television: A Testbed for Multimedia Information Filtering

Arjen P. de Vries
Centre for Telematics and Information Technology
University of Twente

Abstract: We set up an environment in which a database is filled with television data from several information channels. Philips Sound and Vision provides a prototype television that is very suited for this application. The combination of television and personal computer is used to investigate new paradigms to enhance the traditional television interface. Multimedia database technology provides the basis for integration of the program guide in the television interface and the automatical recording of potentially interesting programs. The television helps the user to cope with his information overload. The experimental implementation reveals improvements for the development of the intelligent television. Moreover, the system provides for a testbed environment for multimedia information filtering.

Keywords: multimedia, information filtering, multimedia databases, television, information overload.

1 Introduction

Consumer electronics like television sets contain more and more hardware and software. A high-end television uses over 2 Mb of embedded software. In combination with additional hardware like a set-top box or CD-I player, a television becomes an internet browser. Simultaneously, personal computers take over more and more tasks of the traditional television set. Both PC manufacturers and consumer electronics companies are interested in each others markets.

Philips has developed the Philips Multimedia TV, a prototype television set that can be connected to a common personal computer using an RS-232 serial line. Every function usually available through the remote control can be controlled by applications running on a personal computer. Moreover, the television can return status information to the computer. The serial connection may be applied to assist maintenance engineers. We use this special TV to explore new ways of interfacing with the television set.

2 Problem statement

How often did you just miss that important news bulletin? Who has never been searching for a video tape of some documentary that people mentioned during lunch? Many television and radio channels are broadcasting thousands of programs a day. Even in a relatively small country like The Netherlands we already have seven different television channels and this number will probably continue to grow. However, we do not have the time and capabilities to process all these data. How to deal with this problem of *information overload* is still an open problem.

Electronic television guides will soon be available in several forms. The European standards making body ETSI, in cooperation with the European Broadcasting Union EBU, set a standard for the *Electronic Program Guide* (EPG) [Eur]. EPG is a standardized protocol for a TV guide using electronic data transmission over Teletext. Moreover, a commercial web-site EuroTV provides the program information of over seventy different channels [Eur96]. We expect most broadcasters to deliver information about their programs on the web as well.

We can use a database to manage this program guide information. A modern database system can also store the television data itself. By processing the television data, we produce other representations of the data that can be used for retrieval and filtering tasks [WKSS96]. In combination with the information available from teletext and the data in the electronic program guide, we have a reasonable amount of content information.

We introduce the term *intelligent television* to address the combination of the personal computer and the prototype television set. Using the additional functionality provided by the computer, the intelligent television can talk to the database and help the user deal with the overflow of information. First, the television can automatically turn itself on when an interesting program begins. Never miss the news again! If we express our interest in user profiles, the intelligent television can help us decide what programs to watch. Relevance feedback will help us refine our standing queries [All96]. Finally, the television can use the database's storage functionality to provide VCR functionality. The system temporarily keeps programs matching with our profile until we have time to watch.

Because it is hard to envision new interfaces to something so well known as the television without experimenting with some prototypes, the main purpose of this project is to create a test environment to demonstrate a useful application of an intelligent television.

In the MIRROR project [MIR96], we research new requirements on multimedia databases. Information filtering and retrieval of television data is an application area of multimedia databases [dV96]. We use the Philips prototype television to create a multimedia filtering testbed. This testbed will be used to study the user requirements on multimedia database systems and experiment with multimedia query processing.

3 Design

3.1 Functionality

The primary goal of developing this system is to get insight in the practical consequences of coupling an intelligent television with a database system. We identified two basic tasks that our experimental system should perform:

- remind the user of programs previously selected from the program guide and provide VCR functionality;
- suggest potentially interesting selections using user profiles.

First, the intelligent television provides flexible access to an always up-to-date electronic program guide. The interface to this guide should enable the user to easily create a personal selection from the broadcasted programs for that day or week. He can choose to watch the program. The television checks for conflicts with other selections, and will switch automatically to the right channel when the program starts, even when the television was standby. The other option is to have the program recorded and watch it some other time.

Second, we want to suggest programs to the user. The user first enters an initial description of his information need. An information retrieval process on the descriptions in the electronic program guide then retrieves the program information that may be relevant to the user. The television can suggest another channel while the user watches television. If the television is turned off or standby, the intelligent television may decide to record the program automatically. For this second user task, we need to extend the traditional interface of a television set with some buttons for relevance feedback. The user profile should be refined using this relevance feedback information.

3.2 System architecture

Figure 1 shows the basic architecture. We have an intelligent television composed of a personal computer and a television set. Philips developed this television set for research purposes. The 29" TV screen can be connected to a personal computer running Windows 95. Apart from the normal functionality that a high-end television provides, the prototype TV can display VGA and Super VGA screens from the computer and send and receive commands over a RS-232 connection.

Through the RS-232 connection, PC applications can switch channels, put the volume higher and lower and choose teletext pages. The TV returns information about volume and properties like brightness. Active-X objects hide the communication details from the user applications. To store television data, we connect the television to a computer

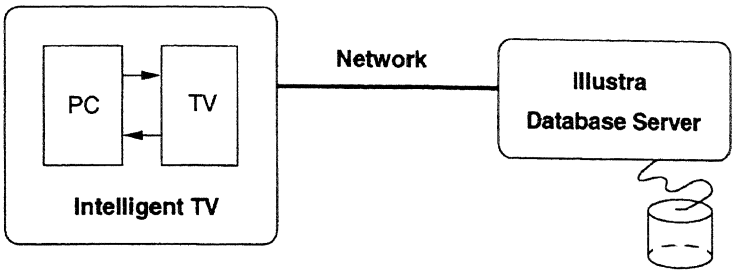

Figure 1: The basic architecture

containing a Q-Motion capture board. The capture device captures Motion JPEG files from the incoming video signal. We can only capture data of the channel the user is watching because the capture board does not have a separate tuner. Adding another tuner would resolve this minor problem if we can control the tuner from the software.

The database provides the key functionality of the design. Illustra Server 3.2 is used to store both captured video and the electronic program guide. Illustra is a commercial object-relational database system. We used the text datablade version 1.2.3 for our experiments. Datablades add new data types and methods to the engine. Other commercially available datablades include image retrieval, spatial data, and document conversion datablades. The database server runs on a SUN Ultra 1 in a local Ethernet network.

We realize that a workstation with a multimedia database is not the kind of equipment we usually find in living rooms. However, we can expect content providers like cable television companies to provide such machines serving small regions. We may move towards the situation in which the content provider stores all broadcasted programs. However, this would require very much storage capacity and network bandwidth. Video on demand experiments have not been very successful. We rather believe in a situation similar to internet providers services. A user buys some space on the server machine that can be used to temporarily store recorded programs, either by user request or by automatic filtering. Local storage devices or compact discs or tapes can be used to replace traditional video tapes for longer duration.

3.3 Conceptual database design

Conceptually, we have an electronic program guide and a multimedia library. The electronic program guide contains the date, the name, the channel that broadcasts it and the starting time for each program. The electronic program guide also provides a short textual description per item. Inspired by the Eurotv web-site, we distinguish between categories movie, series, sports, game, children, news, informative and other.

The multimedia library stores the recorded programs. A recorded program may consist of several parts. Relatively simple algorithms can identify keyframes in video frag-

begin_time	end_time	channel	name	category	description

- *Videoprogram under Program:*

video

- *Part:*

begin_time	channel	subject	part	begin_number	end_number

- *Keyframe:*

begin_time	channel	number

4 Implementation

Because we use multiple platforms and the clients connect to the database server over the network, we decided to do all programming in Java. First, we discuss implementation issues with respect to the database. We describe two datablades, introduce the necessary intermediate server and give some example queries. We conclude with a short overview of the client application and user interface.

4.1 Using datablades

Additional datablades extend the Illustra database with new data types. Our application requires two special data types. First, we need to define a video type to use the database for VCR functionality. Second, we handle text documents describing the programs in the electronic program guide.

4.1.1 Video datablade

To enable clients to view videos, we implemented a rather limited video datablade. Datablades are written in the C programming language. In the datablade, we add a video data type together with some functions to display a video to the server.

Initially, we used the X Window system in combination with the public domain `xanim` viewer. We set the display to the client machine with the function `video_set_display`. Next, we copy the file to disk on the server machine and start the viewer with the correct display value in function `video_display`.

The server will get heavily loaded if many clients select videos this way. Moreover, the X Window protocols have not been developed for displaying video streams. Therefore, we changed the `video_display` function to a function that sends the video to the client. The client starts a video receiver and gives the server its internet address and the port

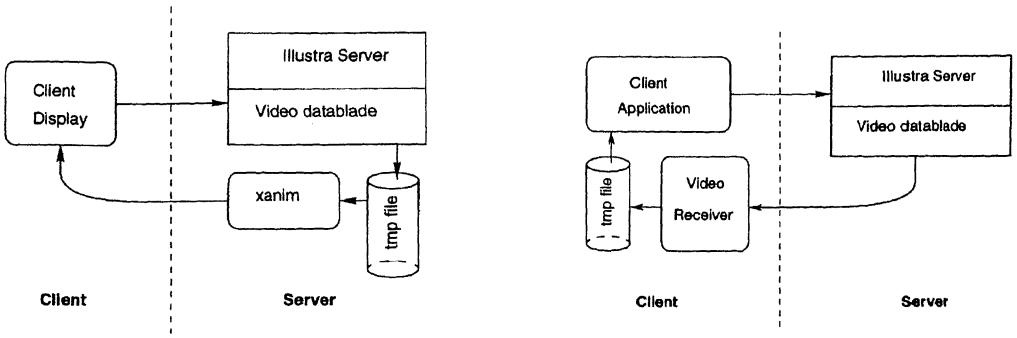

Figure 3: Video datablade in (a.) experimental X Window implementation and (b.) final implementation

number by calling the `set_client` function. In the second step, `send_avi` delivers the video to the client machine. Finally, the client starts a viewer to watch the video.

4.1.2 Text datablade

We model the problem of suggesting potentially interesting television programs to the user as an information retrieval problem. [BC92] argued that information filtering can be described as a special case of the more general information retrieval problem. We restrict ourselves to the retrieval of those descriptions from the database that are similar to the user profile. This problem has been studied extensively for full-text retrieval systems [vR79]. Techniques from text-retrieval have previously been applied to the similar problem of filtering USENET news [YGM95], [SvR91].

The Illustra text datablade extends the engine with text similarity search. The text datablade defines the `doc` data type. Associated to this data type is the function `WeightContainsWords` that determines the relative frequency of occurrence of a word or words in the specified document. If some user remembers a review in a magazine and wants to know whether such a movie will be broadcasted, the following query can help to find the movie he vaguely describes:

```
select begin_time, channel, name,
from only(Program)
where WeightContainsWords(
  'desc_idx', ctid, 'Thriller starring Hugh Grant as young doctor') > 0.7;
```

Unfortunately, using the text datablade caused several problems. Test results indicated that some words were recognized in the description documents and some other words would be ignored. The function `WeightContainsWords` uses a `dtree` index on description. As expected,

```
select * from Program;
```

does retrieve the tuples in `Program` and in `Videoprogram`. However, the index on description does not contain the descriptions of table `Videoprogram`. Indexes on tables are not aware of inheritance relationships among tables.

A work around solution was to move `description` to a separate table and create a relationship between `Program` and the new table. Now, we can create an index on `description` in table `Description` without the inheritance problems. The logical design changes as follows:

- *Program:*

begin_time	end_time	channel	name	category	doc_id

- *Description:*

doc_id	description

Adding a text document to the description table can be done with the following SQL command:

```
insert into description values ('nova_161196',
'format(ascii):descs/nova.doc');
```

Experimenting with similarity search proofs that the weighting functions in the text datablade do not work well for short documents. If the descriptions are too short (i.e. several words), the similarity retrieval turns into boolean retrieval. Moreover, we discovered that adding spaces at the end of documents influenced the values returned by `WeightContainsWords`. Apparently, the text datablade needs much longer documents than we use in the program guide setting. We are still deciding on writing our own similarity function on text descriptions or experimenting with another vendor's datablade.

4.1.3 Client connection

We ran into some strange problems trying to use the Illustra C language client interface from Java native code. An identical stand-alone C program that runs without problems generates errors when running from the Java environment. It probably has to do with the fact that the Illustra library is not thread-safe.

To work around these problems, we can either switch to C on the client side or use an intermediate C server between the Java clients and the Illustra server, as shown in figure 4. We choose to use the intermediate server because now we can easily change the clients or put them on different machine running different operating systems. The drawback of this solution is that we introduce another layer between client and database server. Inserting and updating videos and descriptions has become more complex, because the database views the intermediate server as its client. We must remember to transfer the videos and descriptions across the network from the real client machine before we issue the insertion or update command on the Illustra server.

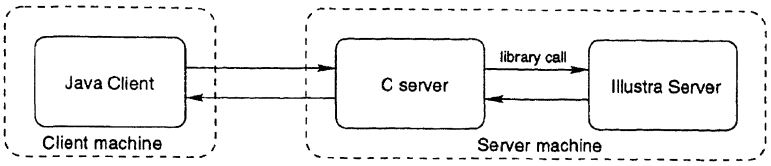

Figure 4: The intermediate server

4.1.4 Querying the database

Formulating SQL commands to select all movies broadcasted this week or the recorded news items on channel *Nederland 1* is pretty straightforward. To request the video 'Nakatomi' recorded from the MTV channel, you use the following SQL statement:

```
select send_avi(video) from videoprogram
where channel = 'MTV' and name = 'Nakatomi';
```

We encountered another problem when we tried to implement object migration. A recorded program has to be removed from table `Program` and inserted into table `Videoprogram`. The video attribute is not nullable, so we cannot first copy the data and then update with the Motion JPEG file. To migrate a program into a video program, the client program has to temporarily store the values of the program's attributes, then delete the object from table `Program` and insert it to table `Videoprogram`. As mentioned previously, we first tranfer a recorded video fragment to the server machine using ftp. Then we are ready to enter the following insertion command:

```
insert into videoprogram values ('1996-11-16 21:00:00',
'1996-11-16 21:10:00', 'Nederland 3', 'Nova', 'informative', 'nova_161196',
'videos/nova.avi');
```

4.2 User interface

The user interface consists of three major units: TV, VCR, and search. TV has the look and feel of a traditional remote control. The user interacts with his intelligent television like with a normal television set.

The VCR mode looks also familiar. The user can rewind, pause and play like with a normal VCR. However, a short list of previously downloaded videos is always present. The code interfacing the capture board uses native code to access the Microsoft Video for Windows API. Because we use the Media Control Interface (MCI), the code is capture board independent. The interface provides full VCR functionality.

The search unit reveals the power of the intelligent television. A simple interface hides the underlying database from the users. Obviously, we do not want the user to enter SQL queries to use his intelligent television. Figure 5 shows the results for a user looking for a

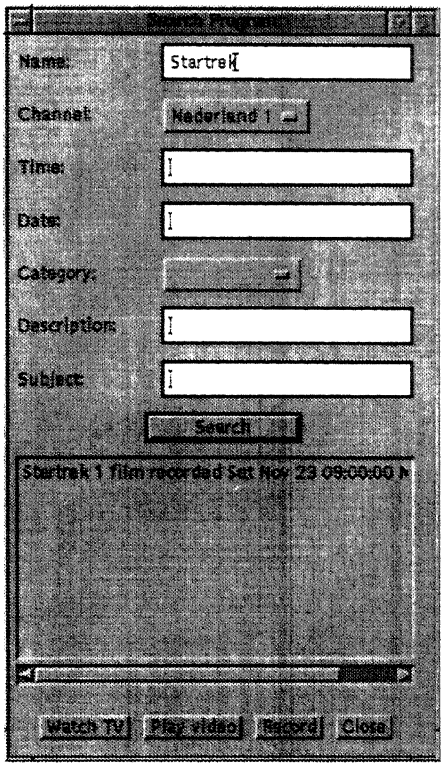

Figure 5: The search interface

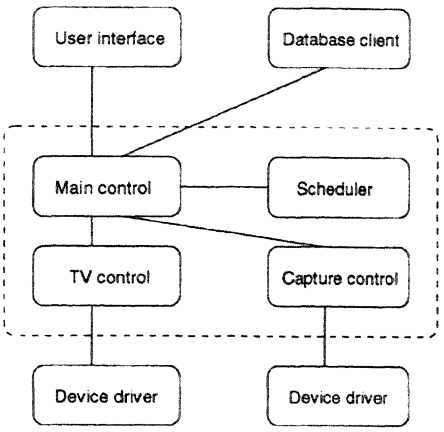

Figure 6: The client application

program named 'Startrek' on channel *Nederland 1*. The movie was recorded on Saturday November 23rd. After selecting the video, the user can play this video on the television.

The intelligent TV controls a play list and a recording list. Had the program not been recorded, the user could have pressed either 'Watch TV' or 'Record'. After checking with the scheduler for conflicts with previously entered requests, the selected program will be added to the play list or the recording list respectively.

4.3 Resource control

Resource control enforces constraints introduced by a limited amount of hardware. Only one program can be viewed at the time, and in the current situation we can either watch TV or record a program.

A task can request access to the TV or the capture board by locking that device. When the task is finished, it should release the lock on the device. We introduced a simple three-level priority scheme to avoid conflicts between different tasks. The highest priority is assigned to Main control (see figure 6). The user interface gets the intermediate priority, and the scheduler the lowest.

5 Evaluation

One purpose of the project was to investigate whether the Philips Multimedia TV is a useful tool to design and experiment with new interfaces to the traditional television.

Browsing the electronic program guide is very powerful. It is a very convenient tool to select the programs to watch. Moreover, after seeing a movie you would like to watch

tonight, you can read the paper on the couch and listen to some music without having to check your watch every other minute. The television will turn itself on when the movie starts.

The intelligent television also enables integration of the television with the VCR. The combination of program guide with multimedia library makes selecting previously recorded programs is far more flexible than with a conventional VCR. We can search on descriptions or just retrieve all videos with the name startrek. However, the search interface is still too restricted. We did not implement the keyframe extraction, so the user can only use name and descriptions to find a previously recorded program. The whole video has to be downloaded before the video can be evaluated.

Developing the system we described gained insight in missing functionality from the prototype television. We suggest two major improvements. First, implementing the system revealed that the prototype television does not return enough status information to the PC. We do not get notified when the user uses its remote control to switch channels or to change to teletext mode. This mixes up state in the client application. Access to teletext would also be very nice, but probably hard to realize over the serial line.

Another big improvement in the television would be overlay of the computer's VGA display on the television stream. In the current prototype, we can only choose between television or computer image. If we had overlay, the extra functionality would be more integrated within the television interface. This enables the intelligent television to first ask the user for permission before switching channels for a scheduled program.

6 Conclusions and further research

We designed an environment using TV, a client PC and a state-of-the-art database server to find out whether information technology can be beneficial for the average citizen. Unfortunately, due to the problems we had with the text datablade, we could not test automatic television suggestions. We believe that the functionality of suggestions would be a main improvement of the system. Although the implemented environment can be improved significantly, the project resulted in the identification of some key issues with respect to further development of the prototype television.

The design project proofs once more that integrating several systems, using different programming languages and advanced technology, is a very tough problem. We experienced many implementation problems, while we just implemented a simple VCR with additional search capabilities. However, we still think that an object-relational database like Illustra is very suited to the application area.

The completed system is used as an initial platform to conduct research into multimedia retrieval. Searching multimedia databases is our primary research interest. The current setup provides automatic data collection from television sources. We will use the

environment to test our ideas on multimedia retrieval in the MIRROR project.

7 Acknowledgments

The author is grateful to Koos de Boer and Joop van der Linden of Philips Sound and Vision for letting us play with the rare prototype TV. Bert Oosterhof of Mosion and the Informix Engines of Innovation grant program gave us the opportunity to work with an Illustra database server. Finally, this testbed would not have been implemented without the help of Wendy Borneman, Mark van Doorn, Jürjen Eisink, Michiel Visser, and Edwin van der Wal.

References

[All96] J. Allen. Incremental relevance feedback. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 270–278, Zürich, Switzerland, 1996.

[BC92] N.J. Belkin and W.B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[dV96] A.P. de Vries. Television information filtering through speech recognition. In *Interactive Distributed Multimedia Systems and Services*, pages 59–69, Berlin, Germany, March 1996. Springer.

[Eur] European Telecommunications Standards Institute. *Electronic Program Guide(EPG); Protocol for a TV-guide using electronic data.*

[Eur96] http://www.eurotv.com/, 1996.

[MIR96] http://www.cs.utwente.nl/~arjen/mmdb.html, 1996.

[SvR91] M. Sanderson and C.J. van Rijsbergen. NRT: news retrieval tool. *Electronic Publishing*, 4(4):205–217, 1991.

[vR79] C.J. van Rijsbergen. *Information retrieval.* Butterworths, London, 2nd edition, 1979.

[WKSS96] H. Wactlar, T. Kanade, M. Smith, and S. Stevens. Intelligent access to digital video: The informedia project. *IEEE Computer*, 29(5), May 1996.

[YGM95] T.W. Yan and H. Garcia-Molina. SIFT - a tool for wide-area information dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 177–186, 1995.

[ZKS95] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar. *A Guided Tour of Multimedia Systems and Applications*, chapter Automatic partitioning of full-motion video, pages 338–355. IEEE Computer Society Press, 1995.