# Exploiting Interactive Video and Animation in Distributed Environments for the Design of Hypermedia and Graphical User Interfaces

CESAR A.C. TEIXEIRA[1]
PETER F. LINNINGTON[2]

[1]Universidade Federal de S. Carlos
Departamento de Computação
São Carlos - SP

[1] Computing Laboratory
University of Kent at Canterbury
Canterbury - Kent - UK
cact@ukc.ac.uk

[2] Computing Laboratory
University of Kent at Canterbury
Canterbury - Kent - UK
pfl@ukc.ac.uk

**Abstract.** The development of high performance computer workstations with video capabilities allows new styles of application, which require simple yet highly dynamic interaction with users, based on the exploitation of graphical user interfaces. We present in this paper the exploration of a distributed environment, which permits the display of animated image and remote analogue video (in accordance with the *video in a window* paradigm), for use as active elements of a hyperdocument or as a medium for developing user interfaces. More specifically, moving regions in a video or animated image can be seen as anchors with links to other segments of a hyperdocument (another video or animated image segment, for example) or as alternatives to the screen buttons of a graphical user interface that provide, when activated, the execution of the specific task.

## 1 Introduction

The widespread use of workstations which combine features like high CPU performance, large storage capacity and high-performance color displays, together with the current state of communication technology and compression techniques, which allow the interchange of information
among interconnected workstations at a much higher rate than was previously possible, have permitted the conception and development of a new range of applications.

We refer to multimedia applications that depend upon use of the workstations' graphical resources, integrate audio and video with other kinds of ordinary data and take advantage of distributed environments. The use of Graphical User Interfaces – GUIs – is a strong requirement for such applications; in many cases, the GUI is an essential part of the application.

Hypermedia applications, which include animation and video as possible media and allow the definition of moving objects from these media as a source of links to other objects, are applications which would not make sense without a GUI. A click over a character of a film, or an atom of a molecular animation can activate, or select, a pre-defined link, allowing access to the information associated with the link destination. Applications of this kind can be constructed from a toolkit which allows anchors to be associated with moving regions of the screen.

The concept of selection of moving objects in hypermedia can also be applied to the development of GUIs, even for conventional applications, in a context where moving images (video or animation) are more powerful than written information to describe an object. The simple user-selection of an option could, in some cases, be much more efficient and user friendly if done over a video or animation sequence than over a complex descriptive and hierarchical menu. A video sequence of a ballet performance, for example, would be more illustrative than a text menu for presenting to the user a set of options of movements, from which the user can choose one to get more information. As another illustration, a sequence of animation that simulates a flight along a route on a bit-mapped street plan would be helpful, from the point of view of localization, and a good alternative to present places to be selected.

The critical parameter, when developing GUIs or hypermedia applications based on moving images, is time. This situation becomes more complex when working in a distributed environment; a large number of questions, related to the synchronization of different media, must be considered.

The development of multimedia applications in a distributed environment not only expands their potential, but also allows the sharing of large amounts of data and expensive devices, and gives access to more processing power to carry out the time-dependent operations involved. Distributing tasks among several process, preferably running in different processors, is a good solution to possible problems on presenting the information to the user at the proper time.

We present in this paper an exploration of a distributed environment, using the ANSA platform [APM, 1992]. The system constructed permits the display of animated image (bit-mapped or generated by an algorithm as a function of the time) and remote analogue video in accordance with the *video in a window* paradigm, for use as active elements of a hyperdocument (see the definition below) or as a useful medium for developing GUIs for a wide range of applications[1].

In this work we have given primary attention to the aspects of hypermedia, since, in the context of our aim of providing mechanism that allows the use of moving-image as a basis for interactions, building GUIs with moving-images for ordinary applications would be a special case of hypermedia application.

For the purpose of this work we consider a hyperdocument as an entity composed of interrelated and linked information, represented in different media, which includes video or animation. The information unit is a segment of the hyperdocument and is called a node. Normally the node contains, or can itself be, the source of links to other nodes; every target of a link is thus a node. A hyperdocument is therefore a web relating nodes to allow navigation; and a node of animation or video information, which we call a moving-image node, is just the data structure that represents an image sequence.

The interactivity of hypertext documents is extended to the moving image nodes of a hyperdocument as well. It must be possible, then, to establish links from a moving-image node (or even part of it) to another node, which can be a piece of text, a graphic, a sound stream or another moving-image node. The "reader" of the document should be able to navigate over the document, selecting not only text screen buttons but also video screen buttons. Since the information of a hyperdocument is not only read, but also interactively read, heard and watched, we prefer to adopt a broader term to identify the user who interacts with a hyperdocument; we call him/her an *"interactive agent"*".

The present state of the technology does not permit a complete exploration of the hypermedia capabilities. Problems related to data compression [Le Gall, 1991], [Wallace, 1991], [Liou, 1991], storage [Sherman, 1988] and networking [Liebhold and Hoffert, 1991], which arise as a consequence of the huge amount of data necessary to represent even a short segment of continuous image, and from the need to use communication paths able to carry large amounts of data with low variance delay [Teixeira, 1989], are not satisfactorily resolved. Some commercial products are already in development but still facing storage and image-resolution difficulties [Green, 1992], [Bruno, 1988]. Synchronizing different media is also a problem, when working in a distributed environment.

The main objectives of this paper are to present the approach adopted to build a distributed system for research purposes on hypermedia, GUIs and other interactive moving-image applications; to describe a mechanism of markup for specifying moving regions in a video or in a bit-mapped animation; to relate and make considerations about the results obtained from the experiments realized.

In section 2 we present the concept of interactive video objects, the difficulties of implementing them in a distributed environment and the solutions adopted.

A markup tool, developed with the purpose of being a user friendly mechanism for authors specifying moving image objects, is described in section 3. The process of synchronizing analogue video with coordinates of image objects is described in section 4.

The analysis of experimental results and the conclusions are presented in sections 5 and 6 respectively.

## 2 Moving Interactive Visual Objects

Having video or animation as an active element in a hyperdocument means that an interactive agent must be able to reach some segment of the hyperdocument by navigating through a link which originates in a video node. The terminal points of a link in a hypertext are commonly called *anchors*. We prefer using this term here with the same abstract meaning as used in [Furuta and Stotts, 1989], and to restrict it for the link source terminals only. That is, an anchor is a part of a node (or the whole node) which contains the information that is related to the target of the link but it is not necessarily its source (the activation-sensitive

[1] At present, the video information is carried and switched in its analogue form. A video server digitalizes and processes this information, in accordance with the position the video must occupy in the user screen, and puts it back in the analogue form. The processed signals are then merged with the analogue signals generated by a workstation and the video is displayed in the chosen position at the monitor screen [Linington, 1988], [Linington, 1990], [Tripp, 1991] . A transition to digital compressed video over an ATM network is scheduled for the end of 1993.

region). So, if an anchor is not the source of a link then an alternative element must be taken for this function; a graphical button in a separated displayed palette is one solution. We define the term *active anchor* for those anchors which are the source of a link as well. An anchor can actually be associated with sources of different links. Since a moving-image consists of dynamic information that changes continuously as a function of the time it is also (or can be) the link associated with the anchors i.e., at each instant an anchor can have one different link associated with it, which leads to a different target node or to a distinct node entry point.

The establishment of anchors in a hyperdocument is an essential task for its author. The simplest way of doing this, over a moving-image node, is to consider the whole image sequence associated with the node, or intervals of it, as a unique active anchor. Although easy to implement, such a policy is very restrictive for applications, since it is not possible to pinpoint the specific region of the image that is more related to the target link; moreover no more than one link can be associated with each image sequence interval.

Another possibility is to create temporally screen-buttons which are associated with the anchors of the moving-image. This approach enables the creation of several passive anchors for the same interval of the image sequence. The problems that arise are the identification of the buttons (which video region is associated with each button?) and the probable visual pollution due to this duplication of information.

A third alternative, which is the one adopted in this work [2], consists of the implementation of active anchors. The problem of screen-button identification does not exist in this case, since the anchors are active; nevertheless, the anchors themselves must be able to be identified to the interactive agent.

We define an object, called *moving interactive visual object – mivo*, that is a region G in the plane (monitor screen) which is determined, at each moment of the image exhibition, as a function of the current frame being displayed. In other words, the region is a function of the time $t$, which is taken with reference to the beginning of the image sequence and can be temporally frozen, increased or even decreased at any rate, depending on the movement conditions of the playback. The function is established by the author in such a way that the region G matches a consistent active anchor at each instant i.e., while the anchor is being displayed during a playback. Therefore *mivos* are objects the existence makes sense only during real time display.

Some restrictions are imposed when the objects are too small or when they move fast on the screen. Probably that is not the situation for most of the image sequences appropriate to be a moving-image node of a hyperdocument, or to be used for building GUIs. Nevertheless, if it is the case, a hybrid solution, using fixed screen-buttons as a source of links as well, could be adopted.

Two well defined phases are carried out to implement *mivos* . The first one is the specification of the regions covering the objects of interest, over a sequence of digitalized image samples. This is an author's preparation activity done when building a hyperdocument or developing a GUI. Using the information generated in the first phase, the second step, which happens at execution time, consists in identifying and transforming, into click-sensitive areas, the regions over the continuous image being displayed, maintaining a correspondence between the coordinates of the regions and the image objects that must be properly covered – this is not a straightforward task when the image sequence is analogue video. An interpolation mechanism is required and suitable synchronization must be sustained.

Since a moving-image object can change its size, shape, and even sometimes its color, an automatic approach of pattern recognition is not suitable for the region specifications carried out during the first phase. Unfortunately human intervention is essential in performing this task properly; it is the responsibility of the author to point out, in each sample, the position of the objects of interest.

We developed a tool to help the author in this task. As we assume that anchors in a moving-image node do not change quickly, in order to make viable its utilization in a hyperdocument or in a GUI, the author needs only to provide a markup over a sequence of frame samples of the moving-image node, instead of doing it for every frame. The samples, in the sequence of frames, can be separated by intervals of about half a second to several seconds, depending on the degree of image change observed in the video node. The result generated by the markup tool is a file with the coordinates of each anchor for each frame of the sample sequence – we call it a *markfile*.

A specific component of the system takes the coordinates of the anchors to generate, and optionally to display, the outlines tracking the corresponding image objects, during the playback. An interpolation algorithm generates the outlines for the frames located between each pair of frame samples produced during the markup procedure.

Animations generated automatically by an algorithm can have the anchors' coordinates generated automatically as well, in accordance with the author's choices and orientation. These coordinates are stored in a file, like the one created by the markup tool, to allow the final phase of *mivos* creation to be realized by the same process as is used for sequences of video or bit-mapped animation.

[2] For moving-image nodes composed of video or bit-mapped animation – not for animation automatically generated by some known algorithm.

The next two sections describe some implementation aspects of the markup tool and difficulties involved in the synchronization process of video and object coordinates.

## 3 The Markup Tool

The tool uses the Athena Widget Set, an X11 facility, for user interfacing. The toplevel window of the user interface keeps a window for image displaying, called the canvas; two windows for object name lists [3]; several small windows working as screen buttons; and some other functional windows described in [Teixeira, 1992]. (see figure 1).

The outlining process is carried out over a sequence of still image samples which were taken, digitalized and stored previously [Ning, 1992b].

The author asks for the exhibition of the digitalized still images on the canvas, chooses names for the objects to be outlined and provides the markup by drawing one or more rectangles over them (in each sample) looking for a suitable coverage.

There is an automatic option which permits the generation of markup rectangles for the present displayed image, based on information taken from the outlines realized on the previous image samples for the same objects. The markup rectangles can be destroyed or modified by the author at any time.

The user has many alternatives for searching for and displaying images in the canvas. These are: activating specific buttons, of which functions are summarized in table 1; manipulating a scrollbar which allows the exhibition of an image from a position in the sequence of samples, corresponding to the position of the bar indicator; pressing the mouse right button – which moves to the next image in the sequence.

Creating an object means choosing a name for a figure or character to be outlined. User friendly drawing facilities were developed to help the author with the task of specifying the region, on the canvas, occupied by the object. By pressing some buttons, and dragging the mouse properly, it is possible to draw, to modify and to delete rectangles of appropriate size and suitable location, with the purpose of framing the desirable region.

Objects may be referenced by their numbers, as an alternative to their names. The numbers are assigned to the objects in accordance with the order of their creation.

The object names/numbers are kept and displayed in two lists: Last Reference List and Numerical or Alphabetical Order List. The Last Reference List displays the last seven objects selected or referenced; each time an object is selected, its name is put on the top of this list, making the task of frequent selections of object names easy when outlining a set of different objects which appear in a scene. The Numerical or Alphabetical Order keeps all the object names, which can be visualized by means of proper manipulation of its embedded scrollbar.

The *manual/automatic* screen-button is a toggle type button. When the exhibition of the next sample is required and the system is in the *automatic* state, the tool provides the exhibition and executes an extrapolation algorithm, drawing markup rectangles over the new image, trying to foresee the positions and sizes of the rectangles from those *just drawn* on the previous sample image.

The extrapolation mechanism can save on author's time and liberate him from repetitive and boring interventions. It is a very useful mechanism, mainly when outlining image sequences where the objects move regularly and slowly.

## 4 Synchronizing Analog Video with Digital Coordinates

The object coordinates obtained from the markup tool are used to create *mivos* when the corresponding image segment is displayed, allowing the interactive agent to select them and, consequently, to invoke the actions linked to those anchors.

As the image segment is exhibited, a variable, which indexes the register of coordinates in the markfile, has its value increased or decreased, depending on the playback direction, so that the static samples are indexed synchronously with the moving-image exhibition. If the interactive agent clicks in a region on the screen where a figure or a character is being shown at that moment, then the system is able to check whether or not that figure was outlined by the author and, when the answer is positive, to follow the associated link; that is, to execute the proper action for that selection at that time.

Creating *mivos* implies indexing, accessing and interpolating the coordinates of objects, which are stored in the markfile, synchronously with the exhibition of the corresponding moving-image sequence, in such a way that it is possible to associate these coordinates with the objects (figures and characters) viewed by the interactive agent. Thus, an interactive agent's action, such as a click on the screen in a region where the image is exhibited, may result in an anchor selection, if the position of the cursor is inside the rectangles defined by the object coordinates at the moment of the click.

As the coordinates of the objects are specified only for some image frame samples (for video and bit-mapped animation), it is necessary to use a mechanism to predict the coordinate positions during the intervals between samples, which can be as long as two, or even more, sec-

[3] In the context of the markup tool, *object* or *moving-image object* are understood as a figure or character (outlined, or to be outlined) which is identified by an exclusive name. It is an appropriate name for anchors; actually it is what will become an anchor when used in a hypermedia application.

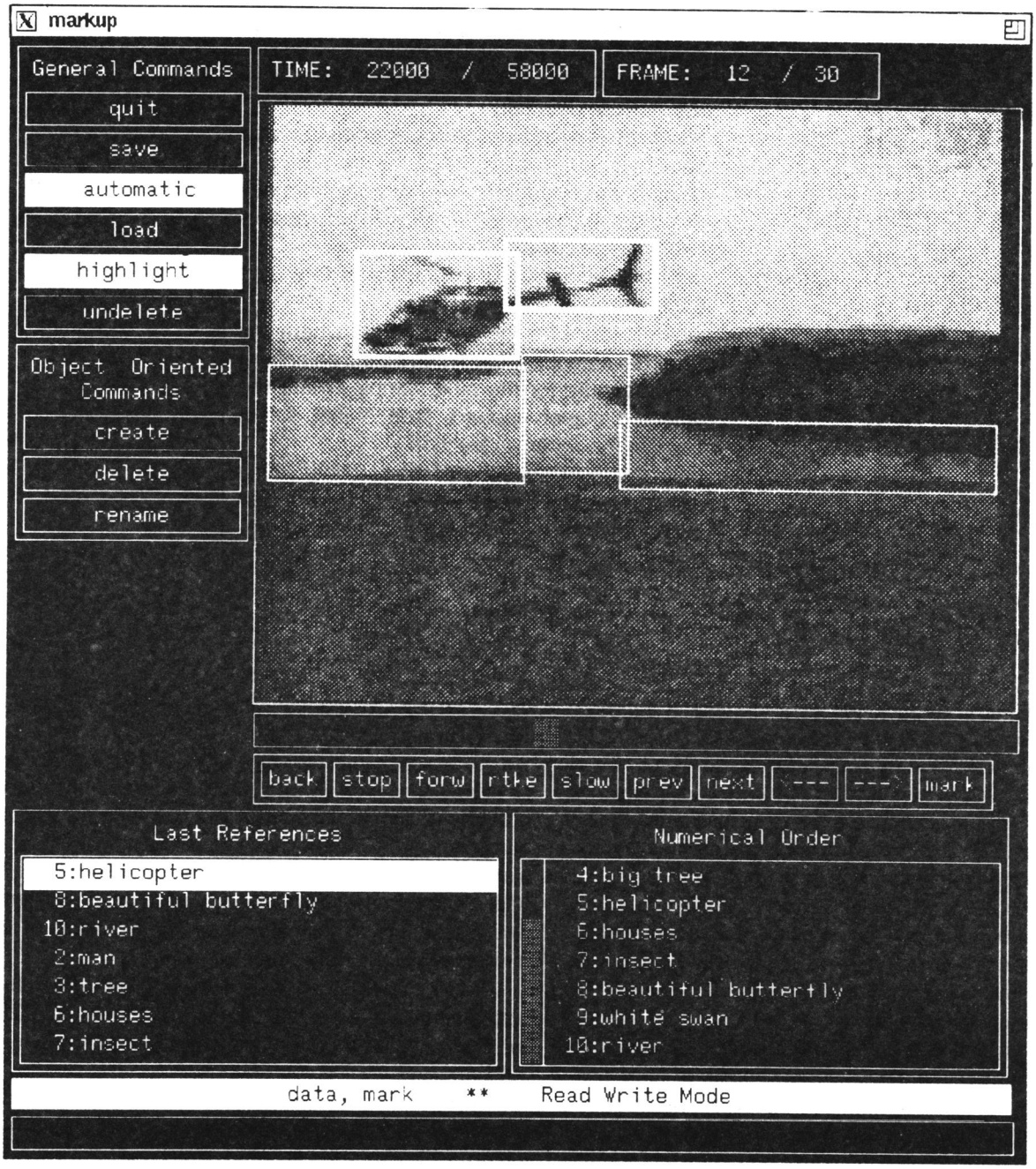


Figure 1: The Markup Graphic User Interface

| button name | button description |
|---|---|
| *next* | displays the next image sample |
| *prev* | displays the previous image sample |
| *forw* | displays continuously the forwarded image samples |
| *back* | displays continuously the image samples in the reverse direction |
| *stop* | aborts the execution of the operations*forw* and *back* |
| *rtke* | restarts the operations *forw* or *back* from five samples, backwarded or forwarded respectively, to the current one |
| *slow/fast* | controls the speed of execution of forw and back operations |
| —> | looks forward for the next sample where the selected object appears (see object selection) |
| <— | looks forward for the next sample where the selected object appears (see object selection), in the reverse direction |
| *mark* | puts a mark in a position of the scrollbar which reflects, proportionally, the number of the current displayed sample related to the total number of samples |

Table 1: Screen buttons for image searching

onds. A linear interpolation algorithm was designed for that purpose; it predicts the coordinates of the objects in real time, while the moving-image is exhibited. These coordinates are generated ever 30 milliseconds, when the moving-image is being played back at normal speed, and can be used as a reference for drawing outlining rectangles around the objects, which keep tracking the figures and video characters on the screen. If there is a click on the moving-image, the coordinates are used to check the occurrence of an object selection; that is, it is verified whether a click has happened inside or outside the rectangles defined by them.

To generate the predictions, the algorithm takes not only the object coordinates for the bounds of the considered period, but also the playback speed and its directions. No interpolation is produced, and consequently no *mivo* survives, while the moving image is presented faster than the normal speed; but the generation of the coordinates and the respective association with the figures are sustained when the image is displayed in slow motion or in the step by step mode. The small response time of the interpolation algorithm is in accordance with the real time character of the application.

We have faced no particular problem when synchronizing animation with the object coordinates, since all the information is digital and under our control. This was not the case when synchronizing analogue video with digital coordinates.

As stated before, the video signals are transported by an analogue network, while information about the video positioning and the coordinates of the objects, among other controlling information, goes through a digital network. The difficulties related to the creation of *mivos* with video arise from the necessity of joining, with perfect synchronization, analogue and digital signals in an interactive agent's workstation window.

To reach this target, we added extra functions to the screen-buttons of a graphic interface, developed at UKC, for remote controlling a video disc laser driver [Ning, 1992a].

When the interactive agent presses the button "play", for example, a message is sent to the video disc laser drive requiring the execution of such a command. We provide a second message asking about the video position at that moment, which means the number of the first frame that will be displayed as a consequence of the execution of the command "play". After receiving that information the interpolation algorithm can start. The big question is to know the time interval bounded by the moment the first frame is displayed, and by the time the coordinates are already processed and ready to allow the generation of the outlining rectangles for that frame.

We observe that the analogue video information is subject to several delays, from the moment the "play" command reaches the processor that controls the video disc laser driver until it is displayed at the interactive agent's workstation. However, since all the resources used are dedicated ones, including the network, which is based on a circuit switch mechanism, the delay variance is practically nil. On the other hand, the digital information — controls from the interactive agent's workstation to the laser video disc processor, and video status travelling in the opposite direction — goes through a digital packet switching network, and is subject to scheduling in a multiprogramming environment.

After several experiments, filming the workstation screen at the moment the video started being displayed, and playing it back in slow motion and in the step by step mode, we could detect an average skew of about one second, with a negligible variance. Therefore, applying a time correction value and an image distortion corrective factor, we obtained a very good synchronization between the objects and its outlinings, resulting in the precise creating of *mivos*.

## 5 Experimental Results and Improvements

We have done many tests to verify the behavior of the system, particularly in relation to the suitability of the linear interpolation algorithm, the matching between analogue images and digital coordinates, and the capability to keep that synchronization.

The linear interpolation algorithm has demonstrated very satisfactory performance, even when tracking objects moving along a non linear route. For these situations, therefore, if the speed of the object is high (but not so high as to make any user interaction impossible), then the intervals between frame samples for the markup tool must be reduced. We obtained very good results adopting intervals of half a second for video nodes with these characteristics. However, in the common situation of scenes presenting nearly stationary objects, a newscaster for example, choosing intervals of two seconds, or even more, has been quite reasonable; this choice saves a lot of the author's work and time with the task of outlining objects.

We have observed that it would be a good practice if the author supervised the process of grabbing images, to get the samples required by the markup tool. He could then decide the rate of grabbing for each scene, depending on the objects to be outlined, and save memory, time and work in the next phases.

Another problem that would be avoided by the author's supervision is the breaks caused by cuts. Two consecutives samples, taken from distinct but contiguous scenes that display the same object but possibly with radically different position and size, will promote wrong interpolation and, consequently, erroneous selections at moments close to the cut. A signal added by the author during the grabbing process, warning of the cut, would also help him to introduce a break in the outlining in the markup phase, and to avoid such erroneous situations[4]. An automatic cut detection, although requiring image analyses, would be more interesting for this task, but even so it would not eliminate the necessity of the author's attendance for the grabbing rate determination.

With the fixed time and image distortion corrections provided, we have obtained very precise *mivos* in our working environment; that is, quite a good match between the object analogue images and their digital outlinings. Nevertheless, a clock synchronization between the interactive agent's workstation and the processor that controls the laser disc driver would be desirable. That would allow the correction of possible distortions, caused by the queue waiting time variance that messages are subject to before being processed in heavily loaded multiprogramming workstations.

Keeping the initial synchronization is not a problem. The experiments proved that taking the number of the first frame only is enough to keep the synchronization for playback periods as long as tens of minutes. Hence, asking for the number of the frame shown each five minutes, for resynchronization interventions, is quite enough to guarantee perfect *mivos* based on analogue videos.

## 6 Conclusions

We developed tools which allow the creation of moving interactive visual objects (*mivos*) in a distributed environment, with the purpose of exploration and design of GUIs and user interactive applications like hypermedia, in such environments. Figures moving on the user's workstation or terminal screen, as a result of a moving-image playback, are conveniently outlined and made sensitive, in such a way that a user selection (clicking a mouse button while the cursor is over the figure, for example) can result in the invocation and execution of an action, linked to that figure for the moment of the selection.

The work was based on a distributed environment which, through the use of an analogue network for carrying video signals, and a video server working cooperatively with other processes to "deliver" the image at the proper position on the screen, makes it possible to offer a service in accordance with the paradigm *video in a window*. Although problems related to the high bandwidth required for video information have been avoided, we had to face and overcome difficulties to synchronize the analogue video image with digital information concerning its

[4]Instead of signaling frame samples, a more laborious, though cleaner, solution would be taking image samples just before and just after the cut.

contents.

The experiments we have carried out, invoking actions from user selections of moving interactive visual objects, through the execution of functions linked to those objects, have shown the potential we have reached for exploring a wide range of issues related to moving-image interactive applications running in a distributed environment.

Finally, it must be observed that any ordinary commercial video disc or animation can be used to build interactive video applications. No special and expensive video disc production/mastering is necessary.

**Acknowledgements**

C.A.C Teixeira has been sponsored by CNPq - (Brazil): grant No 202557/91-0. The group at Kent have been supported by SERC grant GR/F33674 (Palantir).

**References**

[APM, 1992] APM. *ANSAware 4.0 Application Programmer's Manual*. Architecture Projects Management Limited, Cambridge – UK, March 1992.

[Bruno, 1988] R. Bruno. Compact Disc-Interactive. In C. Sherman, editor, *The CD-ROM Handbook*, pages 131–185. McGraw-Hill, New York, 1988.

[Furuta and Stotts, 1989] R. Furuta and P.D. Stotts. A functional meta-structure for hypertext models and systems. *Electronic Publishing Origination, Dissemination and Design*, 3:494–504, June 1989.

[Green, 1992] J. Green. The Evolution of DVI System Software. *Communications of the ACM*, 35(1), January 1992.

[Le Gall, 1991] D.J. Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[Liebhold and Hoffert, 1991] M. Liebhold and E. M. Hoffert. Toward an Open Environment for Digital Video. *Communications of the ACM*, 34(4):103–112, April 1991.

[Linington, 1988] P. Linington. Kent Video Server - Design Objectives. Technical Report 02, Palantir Project - UKC, University of Kent at Canterbury – UK, August 1988.

[Linington, 1990] P. Linington. Notes on the Video Switch. Technical Report 17, Palantir Project - UKC, University of Kent at Canterbury – UK, January 1990.

[Liou, 1991] M.L. Liou. Overview of the px64 kbps video coding standard. *Communications of the ACM*, 34(4):59–63, April 1991.

[Ning, 1992a] L. Ning. A Library to Build Laser-vision Disk Control Panel and its Cooperation with Video Widget. Technical Report 84, Palantir Project - UKC, University of Kent at Canterbury – UK, March 1992.

[Ning, 1992b] L. Ning. Manual Pages on Video Applications. Technical Report 86, Palantir Project - UKC, University of Kent at Canterbury – UK, May 1992.

[Sherman, 1988] C. Sherman, editor. *The CD-ROM Handbook*. McGraw-Hill, New York, 1988.

[Teixeira, 1989] C. A. C. Teixeira. *The Architecture of a Hierarchical LAN for Audio, Video and Data Communication Service Integration*. PhD thesis, University of Sao Paulo, EPUSP - University of Sao Paulo - Brazil, August 1989.

[Teixeira, 1992] C.A.C. Teixeira. A Graphical Tool for Outlining Video Objects. Technical Report 98, Palantir Project - UKC, University of Kent at Canterbury – UK, 1992.

[Tripp, 1991] G. Tripp. Video Controller Processor - the prototype. Technical Report 61, Palantir Project - UKC, University of Kent at Canterbury – UK, May 1991.

[Wallace, 1991] G.K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM*, 34(4):30–44, April 1991.