

# Developing a Multimedia Toolbox for the Khoros System

Yuh-Lin Chang  
Rafael Alonso

Matsushita Information Technology Laboratory  
Panasonic Technologies, Inc.  
Two Research Way  
Princeton, NJ 08540, USA  
{yuhlin,alonso}@mitl.research.panasonic.com

## ABSTRACT

This paper describes our ongoing work on developing a multimedia data processing toolbox for the Khoros system. We first identify the functions commonly used in processing multimedia information. We then discuss how we implement these functions based on the Khoros system data structures and application programming interfaces (API). So far we have developed utilities for format conversion, editing, color conversion, etc., while the implementation of audio and video analysis algorithms is under way.

**Keywords:** multimedia toolbox, Khoros, MPEG, WAV, content-based analysis

## 1 INTRODUCTION

Recent advance of computer hardware technology has made it possible for desktop computers to create and manipulate digital audio/video data. It is therefore crucial to develop powerful and yet easy-to-use software tools for managing the new type of information. This paper discusses our ongoing project toward such a goal. The main objective of our current work is to design and implement a multimedia data processing environment to facilitate future research. In particular, we are interested in the topic of content-based data analysis, which has recently attracted much research interests.<sup>2,9,11</sup>

Our current development work is based on the Khoros system.<sup>5</sup> There are three reasons why we chose *Khoros*. First, *Khoros* can provide many data processing and visualization functions. Second, it has strong support for program development, especially user interface. And, most importantly, it is freely available in the public domain, including source codes.

The functions we developed to support multimedia data manipulation are grouped as a toolbox under Khoros. These modules can be applied directly to process audio/video data or they can be used as building blocks for constructing other functions. We have implemented both data editing and content analysis functionalities in our toolbox. The former includes such functions as view, cut, paste, replace, etc. The latter category contains format conversion, feature extraction, and some other low- and intermediate-level signal processing utilities.

The rest of the paper is organized as follows. In the next section, we introduce the Khoros system and its

features. We then describe our design and implementation of multimedia tools for *Khoros*. Section 4 demonstrates the usage of our toolbox by two examples: one for video processing and another for audio. At the end we discuss the pros and cons of our approach.

## 2 THE KHOROS SYSTEM

Originally developed at the University of New Mexico and currently maintained by the Khoros Research Inc.<sup>1</sup>, *Khoros* is a powerful and yet freely available software system for information processing. The current version is release 1.0.5, while a new version 2.0 is still under development. The major functions in the two versions are (or will be) basically the same. The main reason for re-implementing the whole system is to provide a better software development environment, including things like object-oriented data structure, consistent application programming interface, etc.. Our work is based on the Khoros 2.0 developer release.

*Khoros* has been widely used to solve problems in many different application domains such as image processing and volume visualization. We briefly describe its many features in the following subsections.

### 2.1 Application Toolboxes

*Khoros* aims to solve problems in many different domains. Utilities related to a particular domain are grouped together in a so-called *Toolbox*. The current system comes with the following toolboxes.

- Bootstrap: The bootstrap toolbox contains the utilities for installing *Khoros*.
- Design: The design toolbox includes the GUI and the visualization function.
- Data service: The data service toolbox consists of a collection of libraries for accessing data.
- Data manipulation: the data manipulation toolbox provides library functions for manipulating data.
- Envision: The envision toolbox provides data visualization service such as 3-D display, volume data visualization, etc..
- Geometry: The geometry toolbox contains the utilities for processing and rendering geometry objects such as polylines, circles, etc.
- Image: The image toolbox implements image and signal processing functions such as linear and nonlinear filtering, enhancement, edge detection, etc.
- Matrix: The matrix toolbox provides mathematical functions for handling matrices.

However, to our knowledge, currently *Khoros* cannot directly handle multimedia data because: (1) it is not yet compatible with standard audio and video data formats, and (2) its general-purpose algorithms sometimes are not adequate for processing multimedia data. In our opinion, it will be very useful to empower *Khoros* with multimedia information processing capability, since so many tools are immediately available.

---

<sup>1</sup>Their WWW address is <http://www.khoros.unm.edu/>.

## 2.2 User Interface

Central to the Khoros system is a set of flexible user interfaces. Users can communicate with most *Khoros* functions in three different ways:

- A data-flow oriented visual programming language called *cantata* can be used for algorithm design and fast prototyping . The user builds a *cantata* application program (called a “workspace”) by connecting processing nodes (called “glyphs”) to form a data flow graph. Glyphs can be selected from many library functions available in *Khoros*. In Section 4, we show two such example workspace.
- Functions can also be called from command line or from shell-scripts for batch processing. Complete on-line help files and manuals come with the package.
- Most functions also have a library object version, which can be called from within a C or C++ program. The application programming interface is consistently defined for all *Khoros* library subroutines.

## 2.3 Programming Environment

*Khoros* is not only an application system but also a software development environment. The current version (v2.0) contains the following tools.

- Craftsman: The craftsman is used to create new toolboxes or to modify existing ones.
- Composer: The composer is used, in conjunction with craftsman, to create, delete, or modify software objects (functions) in a toolbox.
- Guise: The guise can be used, together with the composer, to allow a user interactively create or modify the graphical user interface (GUI) for a software object.
- Ghostwriter: The ghostwriter can automatically generates programs and documentations for a software object.
- Conductor: The conductor can automatically generates X-window based GUI for application programs.

# 3 MULTIMEDIA EXTENSION OF KHOROS

This section discusses our effort to empower *Khoros* with multimedia information processing capability. Basically, the visualization functions in *Khoros* are useful for editing data, while the image and signal processing subroutines can obviously be used for automatic content analysis. In the following we first discuss how to import multimedia data to the Khoros system. We then present our extension of *Khoros* that provides both multimedia data editing and analysis capabilities. The functions we implemented can be put into three categories: format conversion, object editing, and miscellaneous utilities. In the next stage we plan to implement more audio and video analysis algorithms using existing Khoros system subroutines.

## 3.1 Format Conversion

*Khoros* uses a data format called *VIFF*, which is a general-purpose data representation. Its most basic form is a five-dimensional array of data (3-D volume position + color + time), as Figure 1 illustrates. The data type can

be byte, short, int, long, float, or even complex. Furthermore, VIFF can also contain color maps and geometric objects such as lines and circles. However, to our knowledge VIFF is not supported by most major multimedia software systems. Henceforth, the first step is to convert other popular multimedia data formats to VIFF. We have implemented two sets of format conversion functions: MPEG for video data and WAV for audio. Once a MPEG or WAV object is converted to the VIFF format, it can then be manipulated by all the existing *KhoroS* functions.

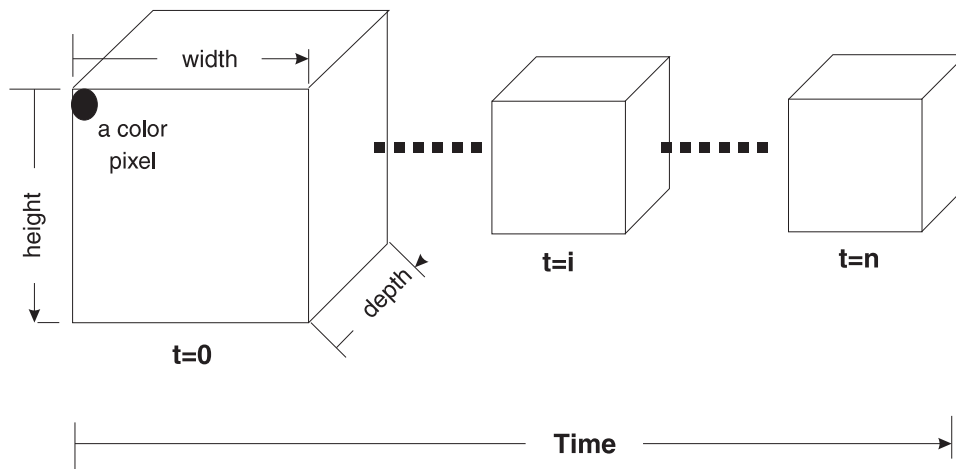


Figure 1: Viff data structure.

### 3.1.1 MPEG Conversion

MPEG (Moving Picture Experts Group)<sup>3</sup> is an international standard for digital video compression. In a nutshell, the MPEG encoding algorithm uses two basic techniques to compress video data: block-based motion compensation to explore temporal redundancy, and DCT (Discrete Cosine Transform) to explore spatial redundancy. Currently, MPEG-1 can deliver VHS-quality digital video with data rates around 1.5 Mbps (mega bits per second), and MPEG-2 can be used to encode HDTV video with data rates from 4 to 10 Mbps.

Our MPEG conversion programs are based on MPEG Software Simulation Group's (MSSG) public domain package.<sup>7</sup>

- **ImportMpeg.** The original MPEG decoding function in the MSSG package can only write to a sequence of single images in the YUV, SIF, TGA, X11, and PPM formats. We modify the picture output subroutine so that an MPEG video file can now be converted to a VIFF object.
- **ExportMpeg.** Similarly, to convert VIFF data to MPEG, we modify the encoder in the MSSG package. The picture reading subroutine has been rewritten to read VIFF data.

### 3.1.2 WAV Conversion

In a multimedia document, the audio tracks usually contain very important information and therefore in our toolbox we intend to integrate audio processing with video. So far we have implemented two format conversion utilities, while the development of audio feature extraction algorithms is still under way.

Among the many different digital audio data formats, WAV seems to be gaining ground because it is backed by MicroSoft. We have developed programs to convert WAV audio data to and from 1-D VIFF data based on a public domain sound processing package called *SOX*.<sup>8</sup>

Ideally, audio should be stored with video as one object to keep the synchronization information intact. Unfortunately, right now we have to store them as separate streams because VIFF does not support multi-track data representation. We will discuss possible solutions to this problem in the conclusion section.

### 3.2 Object Editing

Although inexpensive, commercial video editing and authoring tools are already available, e.g., *Adobe Premier*<sup>1</sup> and *Macromedia Director*,<sup>6</sup> their functions are usually limited and none of them can be easily integrated with content-based analysis tools.

The design goal of our multimedia tool is similar in concept to the idea of *Algebraic Video*,<sup>10</sup> where basic algebraic operations can be applied to create new video streams from existing ones. We have implemented a few functions for editing audio and video objects in *Khoros*. These functions are similar to existing *Khoros* functions in style, i.e., the data-flow model. Figure 2 illustrates a generic audio/video (AV) data processing unit, which inputs one or more AV streams, process them, and then outputs one or more AV streams. Currently we have implemented the following four editing functions, while in the future we may consider to extend them to the more intuitive, time-line based editing environment.

- Cut AV: The cut AV function cuts an audio or video object into two parts.
- Insert AV: The insert AV function insert one audio or video object into another.
- Replace AV: The replace AV function replace part of an audio or video object into by another.
- Mark AV: The mark AV function marks part of an audio or video object.

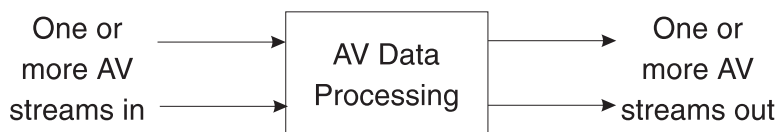


Figure 2: A generic AV data processing unit.

### 3.3 Miscellaneous Utilities

The following functions are developed to help manipulating audio and video objects.

- Decolorize video: The decolorize function converts a color (RGB) video into grey-level. Such a function is needed because some image processing functions, such as edge detection, are better defined for gray-level images.
- Extract raw data: This function extracts and prints out the raw data from a *Khoros* object.

## 4 EXAMPLES

In this section we present two example applications of our multimedia toolbox. The first example demonstrates the processing of video data, while the second illustrates audio signal processing.

### 4.1 Video Processing

The first example demonstrates video data manipulation. Figure 3 shows the cantata workspace, and Figure 4 illustrates the flowchart of operations. This workspace implements a well-known video shot segmentation algorithm based on histogram difference.<sup>4</sup> Basically, a *cut* is detected if a frame's histogram is considered "substantially different" from that of its previous frame, based on the  $\chi^2$  comparison:

$$\sum_{i=1}^G \frac{(H_t(i) - H_{t-1}(i))^2}{H_t(i)}, \quad (1)$$

where  $H_t$  is the histogram for time  $t$ , and  $G$  is the total number of colors in an image.

The functions used in this example are explained as follows. The functions developed by us are shown in *italics*, while the original Khoros functions are shown in **bold face**.

- *Import MPEG* converts an MPEG encoded data stream into VIFF.
- **Animate** is a Khoros function for displaying a VIFF object as an animation sequence.
- *Decolorize* converts a color (RGB) VIFF object into grey-level.
- **Histogram** is a Khoros function for computing the histogram of a VIFF object.
- **Translate** is a Khoros function for shifting a VIFF object in time.
- **Subtract** is a Khoros function for subtracting two VIFF objects.
- **Replace Value** is a Khoros function for replacing the values of a VIFF object. It is employed to avoid division by zero.
- **Square** is a Khoros function for applying the squaring operation on a VIFF object.
- **Divide** is a Khoros function for dividing two VIFF objects.
- **Statistics** is a Khoros function for computing the statistics of a VIFF object.
- *Shot Segment* detects the shot transition boundary by locating the peaks in the histogram difference sequence.
- **File Viewer** is a Khoros function for viewing the contents of a VIFF object.
- **2D Plot** is a Khoros function for plotting 2-D graphs.

We use a football game video with 1471 frames as an example. Figure 5 shows the beginning of this video and the segmentation results. The key frames extracted by the segmentation process are shown in Figure 6.

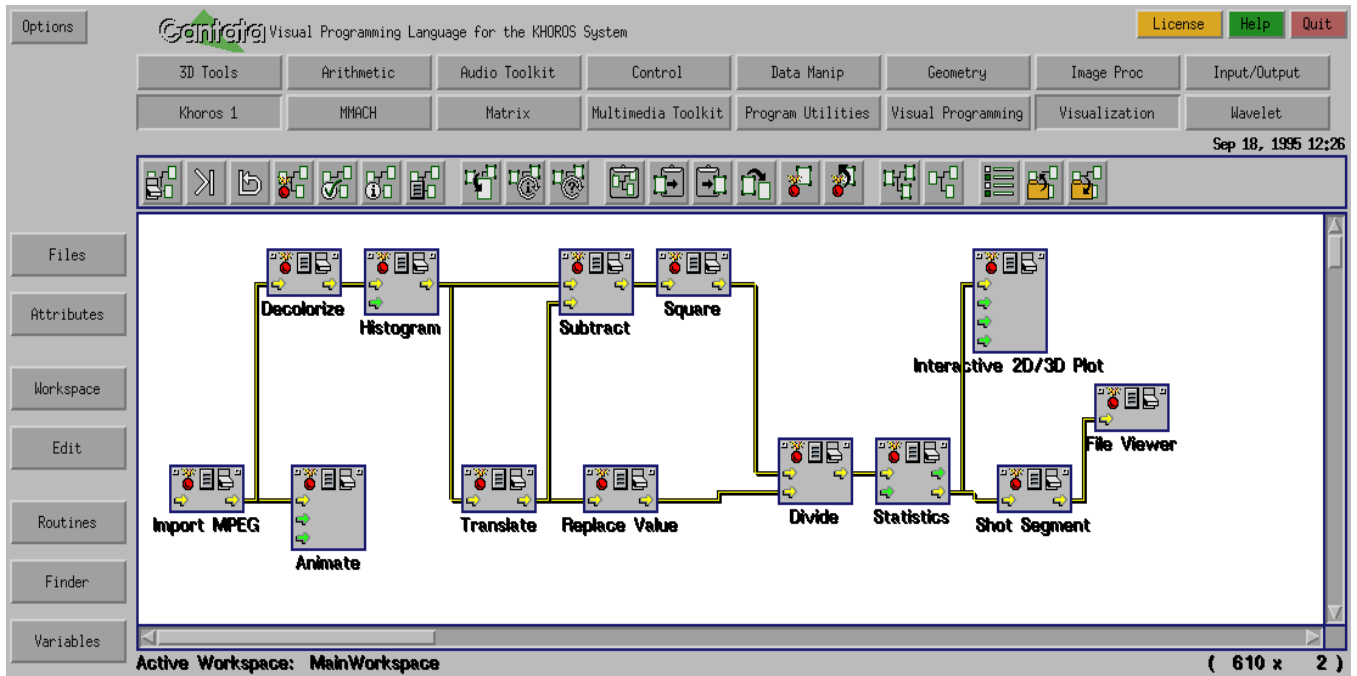


Figure 3: An example of Khoros work space for video shot segmentation.

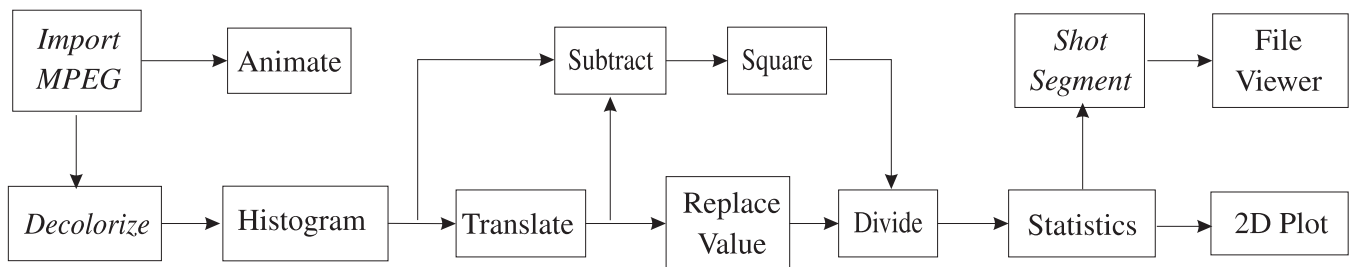


Figure 4: Flowchart of the video shot segmentation algorithm.

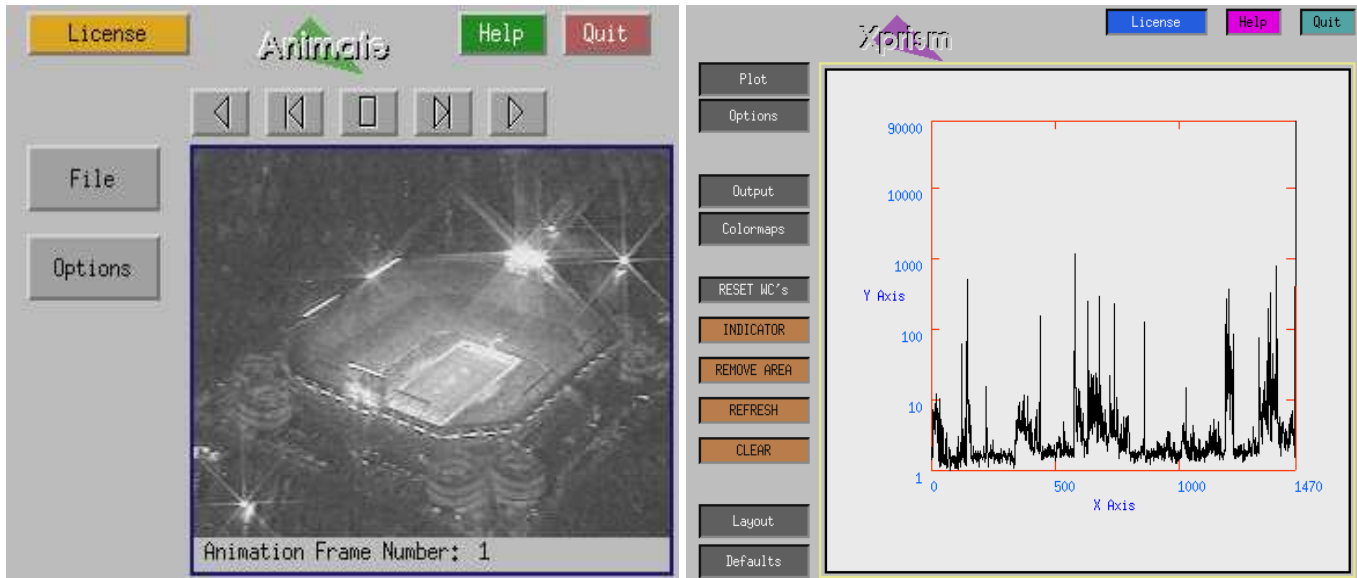


Figure 5: Example football video and the cut detection results.

## 4.2 Audio Processing

The second example deals with audio data processing using the Khoros wavelet toolbox. Figure 7 shows the cantata workspace, and Figure 8 illustrates the flowchart for the operation. This example demonstrates how to quickly set up a prototype system for testing audio compression algorithms based on the wavelet transform. 1-D Daubechies filters of order 2 (four coefficients) with two levels are used in the forward and inverse transform. Figure 9 shows the original audio signal and its wavelet transform. Simple clipping is then applied to replace small integers by zeros. However, the twist is that different subbands can have different threshold parameters according to their perceptual importance or other contextual information. In a real system, more complicated lossy compression methods are likely to be used after clipping. Finally, Figure 10 shows the reconstructed signal and its wavelet.

The functions used in this example are listed as follows. Again, the functions developed by us are shown in *italics*, while the original Khoros functions are shown in **bold face**.

- *Import WAV* converts a WAV encoded data stream into a VIFF object.
- **Normalize** constrains the values of VIFF object to the range 0-255, and converts it from **long** to **byte** for this example.
- **Wavelet** performs the forward and inverse wavelet transform on a VIFF object.
- **Extract Subbands** extracts the subbands from a wavelet transform, so that they can be processed separately.
- **Clip** is used to change small numbers into zeros, so that further compression, such as *run length coding*, can be applied.
- **Append** combines more than one VIFF objects into one.
- *Export WAV* converts a VIFF object into a WAV stream.



Figure 6: Key frames extracted by shot segmentation.

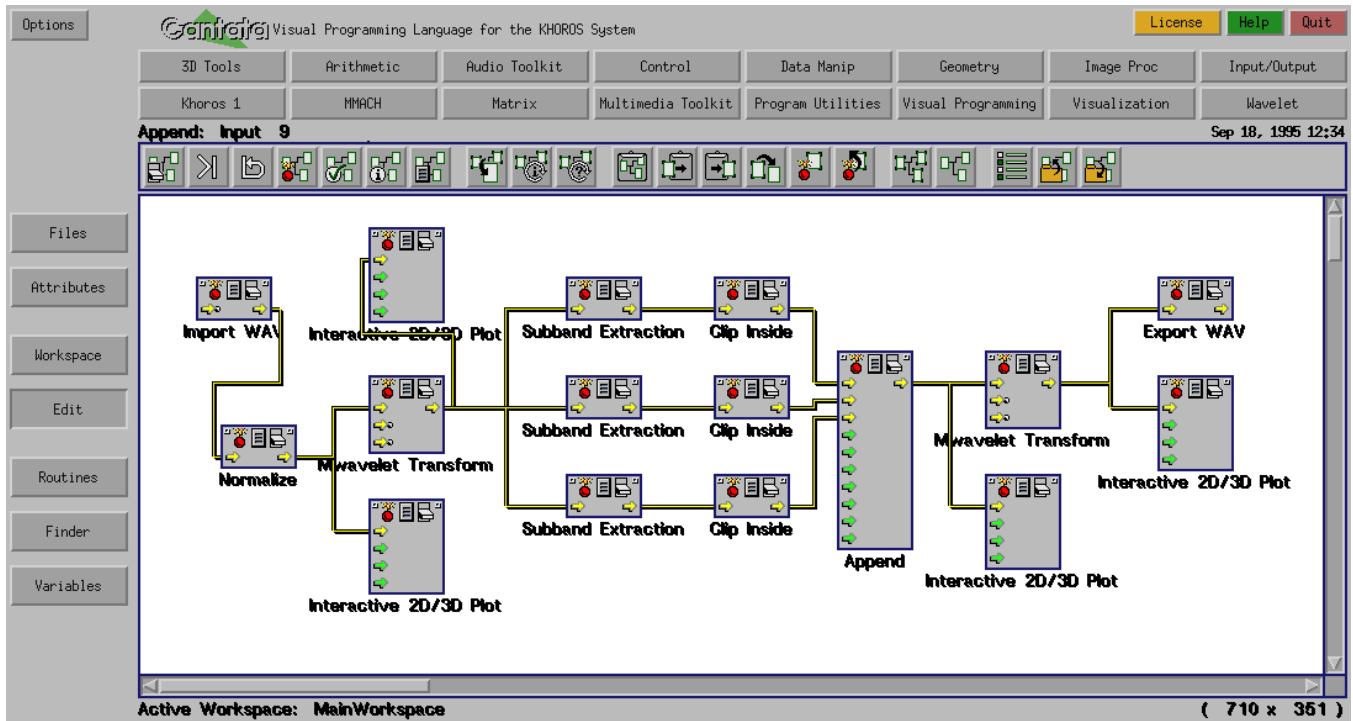


Figure 7: An example of Khoros workspace for manipulating WAV audio.

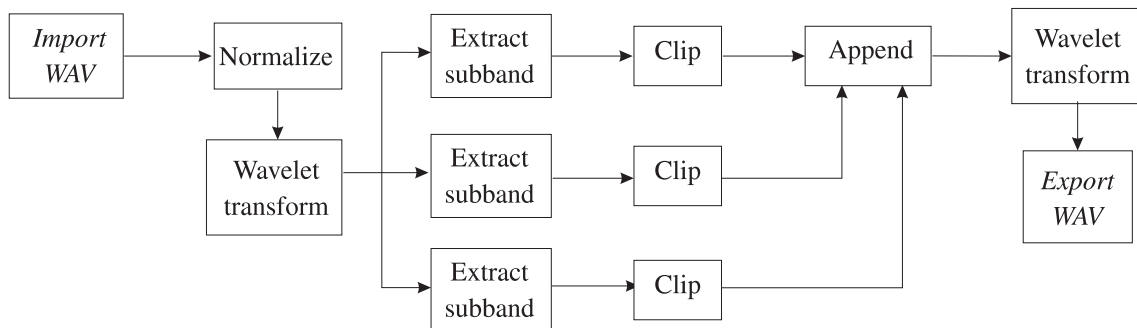


Figure 8: Flowchart of the WAV audio manipulation.

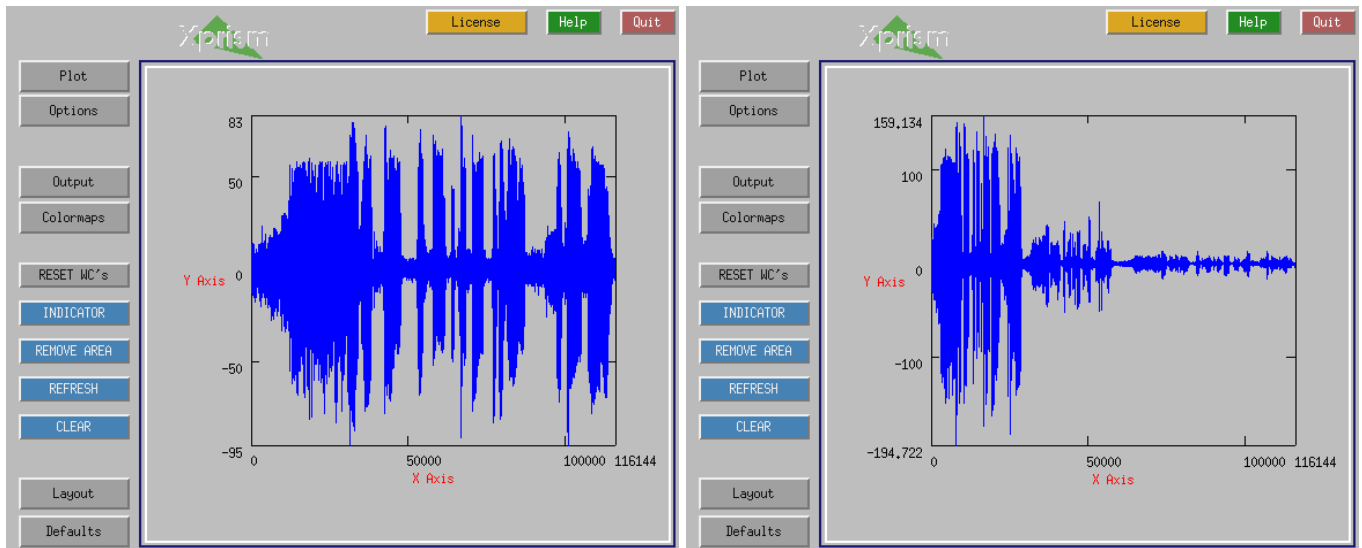


Figure 9: Original audio signal and its wavelet transform.

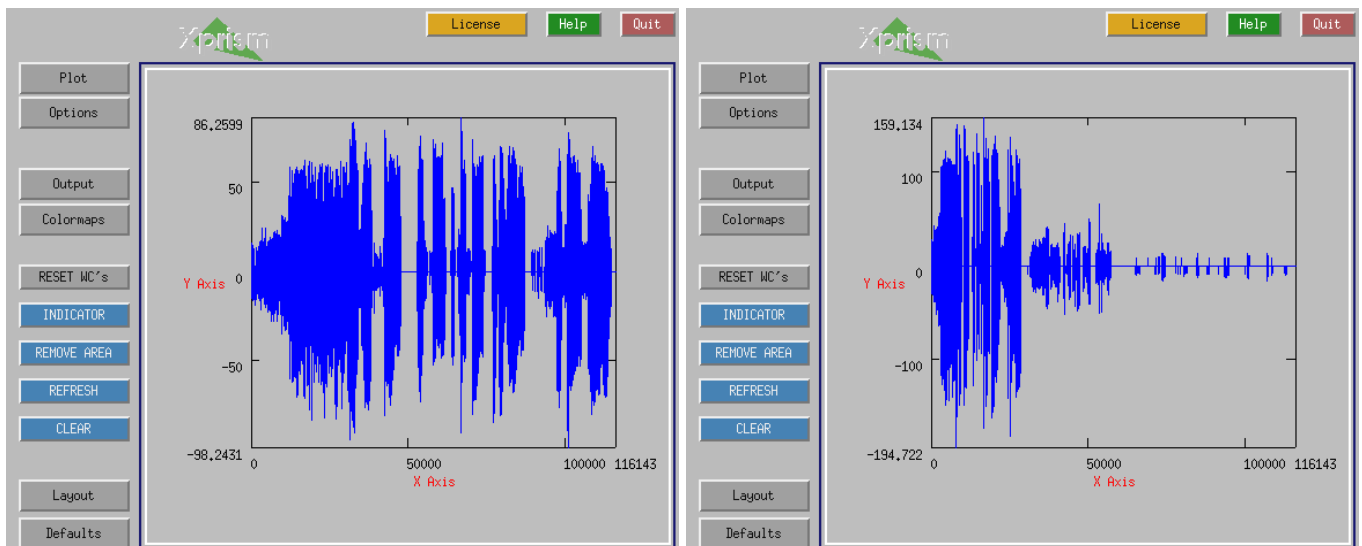


Figure 10: Reconstructed audio signal and its wavelet transform.

## 5 CONCLUSION

This paper discusses our work on developing a multimedia information processing environment based on the Khoros system. We intend to further experiment with content-based audio and video analysis algorithms in such an environment. We chose to base our work on *Khoros* because of its rich set of data processing utilities and its programming environment.

On the other hand, during the toolbox development we also notice a few problems associated with *Khoros* that probably can be improved by further studies and better design.

- Synchronization. Currently, video and audio have to be represented by separate objects, while in reality the two should be stored together as a multi-track object.
- Compression. Right now *Khoros* uses little or no compression at all when objects are stored internally. This causes the I/O bottleneck for large videos.
- Codec hardware. Since *Khoros* has to be run under Unix, it has not taken full advantage of the availability of inexpensive multimedia hardware in the PC world, e.g., sound cards and MPEG playback cards.

One solution to the first two problems is to build a meta-system on top of *Khoros* that includes these functions. However, we believe that a better and easier solution is to convince the *Khoros* development team to consider more multimedia functionalities in their next release. As for the last issue, we are currently contemplating on porting our work from DEC Alpha to the Linux environment on a PC, in order to explore the possibility of directly accessing multimedia hardware from within *Khoros*.

## 6 REFERENCES

- [1] The Adobe Systems Inc., Mountain View, CA. *Adobe Premier 1.1 User's Manual*, 1993.
- [2] A. Akutsu and Y. Tonomura. Video tomography: an efficient method for camerawork extraction and motion analysis. In *The Second ACM International Conference on Multimedia*, pages 349–356, October 1994.
- [3] D. Le Gall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.
- [4] A. Hampapur and T. Weymouth. Digital video segmentation. In *The Second ACM International Conference on Multimedia*, pages 357–364, October 1994.
- [5] K. Konstantinides and J. R. Rasure. The Khoros software development environment for image and signal processing. *IEEE Transactions on Image Processing*, 3(3):243–252, 1994.
- [6] The Macromedia Systems Inc., San Francisco, CA. *Macromedia Director 3.0 User's Manual*, 1993.
- [7] MPEG Software Simulation Group. *MPEG-2 Software Codec*, 1994.
- [8] J. Poskanzer. *SOX: SOund sXchange*, 1989.
- [9] S. W. Smoliar and H. Zhang. Content-based video indexing and retrieval. *IEEE Multimedia*, 1(2):62–75, 1994.
- [10] A. Weiss, T. Duda, and D. K. Gifford. Algebraic video. In *The First IEEE International Conference on Multimedia Computing*, pages 357–364, May 1994.
- [11] A. Yoshitaka et al. Knowledge-assisted content-based retrieval for multimedia databases. *IEEE Multimedia*, 1(4):12–20, 1994.