GUI FRONT-END FOR SPECTRAL WARPING

Mr. T. Healy, Mr. T. Lysaght, and Dr. J. Timoney

Department of Computer Science
National University of Ireland, Maynooth, Ireland
Tom.Lysaght@may.ie,Joseph.Timoney@may.ie

ABSTRACT

This paper describes a software tool developed in the Java language to facilitate time and frequency warping of audio spectra. The application utilises the Java Advanced Image Processing (AIP) API which contains classes for image manipulation and, in particular, for non-linear warping using polynomial transformations. Warping of spectral representations is fundamental to sound processing techniques such as sound transformation and morphing. Dynamic time warping has been the method of choice for many implementations of temporal and spectral alignment for morphing. This tool offers greater advantage by providing an interactive approach to warping, thus allowing greater flexibility in achieving a desired transformation. This application can then be used as input to a signal synthesis routine, which will recover the transformed sound.

1. INTRODUCTION

The most popular method for time and frequency alignment of features in sound processing algorithms has been dynamic time warping (DTW) [1,2]. This technique permits feature matching and alignment through a one-dimensional warping process. However, it is limited in performance as it depends completely on the complexity of the backtracing procedure and thus misalignments can happen easily. Spatial warping of images, on the other hand, offers a two-dimensional approach to time-frequency warping. Furthermore, the development of an interactive tool for warping enables greater flexibility and ease in specifying the warping parameters, thus allowing immediate examination and verification of results or editing of control points. The Java language was chosen in which to implement this application due to the availability of the Advanced Image Processing API which offers polynomial warping routines. Furthermore, Java facilitates the development of a portable GUI interface for such an application. Figure 1. shows a flow chart of the warping process and illustrates the key implementation features. Section 2 outlines the mathematics behind polynomial transformations for warping. Section 3 details the software design and implementation. Section 4 describes the JAVA implementation. Section 5 describes software testing and results. Section 6 draws conclusions and points to future work.

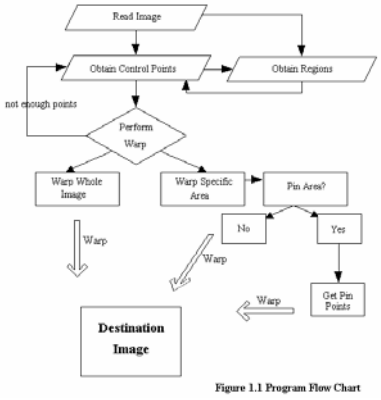

Figure 1: Warping program flow chart.

2. POLYNOMIAL TRANSFORMATIONS

Time-frequency spatial warping is the process of distorting an original/source time-frequency representation into a destination representation according to a mapping between source and destination coordinates termed control or fiducial points [3]. The positions of the coordinates in the destination image are altered according to some warping function. The spatial transformation is expressed in general form as a mapping from a point *(x, y)* in an output time-frequency representation to its corresponding warped position *(i, j)* in an input representation (see Eq. 1).

$$(i, j) = (W_x(x, y), W_y(x, y)) \tag{1}$$

where W_x and W_y are the warping functions. These functions can be modelled by the following polynomial transformations (Eq. 2):

$$W_x(x, y) = \sum_p^n \sum_q^n a_{pq} x^p y^q$$
$$W_y(x, y) = \sum_p^n \sum_q^n b_{pq} x^p y^q \tag{2}$$

where *n* denotes the degree of the polynomial. Second order warping is usually sufficient to effect most distortions. Solving for the a and *b* coefficients in Eq.1 is achieved using an over-determined system of linear equations. Such a system does not have an exact solution and so the common approach of the least-squared-error solution is used. Estimation of the intensity of the output coordinates uses bilinear interpolation when the

corresponding input position yielded by the warping function is non-integer.

3. GUI DESIGN AND IMPLEMENTATION

This GUI was designed to be as user friendly as possible by making it transparent and as consistent with other GUI's as possible. The dialogs were designed and implemented to yield closure and provide informative feedback where required. When designing the interface it was assumed that the user would have a reasonable knowledge of the basic ideas of image warping such as polynomial degree and the number of control points required. The GUI itself is divided into three distinct regions (see Figure 3). The largest part of the GUI is used to display the image loaded by the user. This immediately attracts the user's attention to the main purpose of the application. On the right of this is the toolbar that contains an array of buttons and labels that are used both to control the application and to provide important feedback to the user. The final region of the GUI is the menu bar. This contains standard options concerning file operations as well as advanced options such as a zooming tool.

Each of the dialogs used in the application are modal. This prevents further user input to the rest of the application until the dialog has been dealt with. This feature in addition to informative error messages that are provided to the user form an effective error prevention system. This has the effect of increasing the users confidence in the application.

There are a number of features of the interface that improve the interaction between the user and the application. The first of these features is the changing mouse cursor, which allows the user to easily identify the mode which the application is in. Each cursor represents a different mode and indicates to the user which operations need to be performed before the warp operation can take place. The use of modal message dialogs informs the user of the progress of the warping process and provides the user with options regarding choices which need to be made by the user such as the selecting of pin points.

The menus provided in the application allow the user to access a number of more abstract support features such as file operations, zoom feature and a feature which allows the user to obtain the intensity values of each pixel in the image.

4. JAVA IMPLEMENTATION

Java was chosen as the programming language for the GUI for a number of reasons. The primary reason for using Java was the existence of the Java Advanced Imaging (JAI) Applications Programming Interfaces (API). This JAI API allows for sophisticated, high-performance image processing to be incorporated into Java applications. This enables us to use JAI to perform operations such as warping on the images in the application. The GUI was implemented using the AWT and Swing packages provided by the Java language. As an object-oriented programming language Java allows for greater

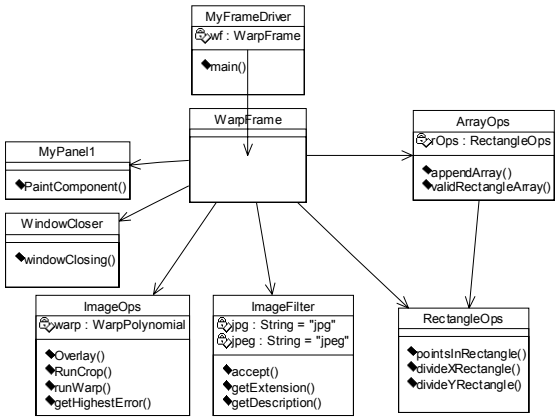

Figure 2: System Architecture Design

modularity and reusability, which is an important feature of software applications. Another compelling reason to program the GUI in Java is platform independence or portability. Java runs on most major hardware and software platforms, including Windows 95 and NT, the Macintosh, and several varieties of UNIX. Java applications also have the advantage of maintaining their 'look and feel' irrespective of the platform on which they are running. Consequently the application, which has been developed, will work as well, for example, on a Macintosh as it will on a PC running Windows software.

The GUI components from the AWT package are linked directly to the local platforms GUI capabilities and so may look and feel different depending on the underlying platform. However Swing is different in this respect as its components are implemented without using native code and consequently are independent on the underlying platform. The AWT contains the classes that handle events, the successful implementation of which plays a central role in the success of interaction between user and interface. This application relies heavily on input from the user in the form of mouse clicks and mouse movement.

Figure 2 shows a UML diagram illustrating the architecture of the application. *MyFrameDriver* is the parent component and allows running more than one instance of the main component. The *WarpFrame* class is the main component of the system. All of the essential variables and data structures are instantiated and updated here. This creates the GUI and supports all event handling that occurs during the execution of the application. Six support components are used by the main component to perform the warping. The first of these, *ArrayOps* is used to carry out operations on the array data structures which store the coordinates of the control points. Another component, RectangleOps is concerned with validating user input and with the implementation of the pinning function. The pinning function allows specification of extra control points or pin points along the edges of the region to be warped in order to avoid unnecessary skewing of the surface. A third component handles all the paint operations such as drawing the rectangular regions which outline the areas which have been selected for

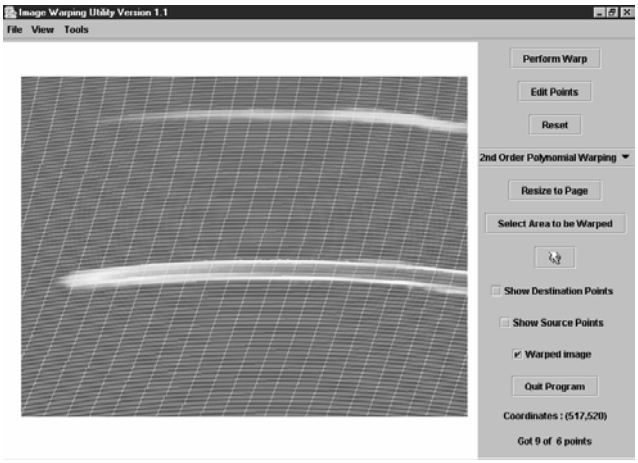

Figure 3: GUI interface and warping example

warping. The component which deals with the various image processing procedures for warping uses a number of methods from the JAI API. Another feature which enhances the user's interaction with the application is provided by the *ImageFilter* component which limits the file types available under the load/save menu. This prevents the user from loading file types which are not compatible with the application namely non-image files.

The application was coded using a mixture of Java Swing and AWT components. The 'top-level' window for the application was created using a Swing *JFrame* object (See Figure 3). Other components added to this include *JButton*, *JComboBox*, and *JMenu*. The AWT package contains classes that provide event handling. For example, the *MouseEvent* class is used to tell the current coordinates of the mouse. These coordinates are visible as a label at the bottom of the toolbar (see Figure 3). Event handling enables the application to respond to certain main modes of operation, indicated by the cursor changing. These include, control point selection/editing. Dialog boxes, *dialogs*, are used to display different *error/question* messages to the user. These *dialogs* are all 'modal' as an error prevention mechanism.

5. TESTING AND RESULTS

Testing consisted of component testing of the GUI and black box testing to test the response of the application to large input sets. Warping was tested using varying control point sets for the different warping orders. Figure 3 shows two harmonics of a violin time-frequency distribution warped at the midpoint along the time axis.

Synthesis of the warped images of Wigner distributions, in particular the smoothed pseudo Wigner distribution (SPWD), was implemented using MatLab. The SPWD magnitudes were written as uncompressed *PNG* image files. The Wigner distribution has no phase information and so only the SPWD magnitudes are warped. After warping, the saved *PNG* warped image was imported into MatLab and resynthesised using a Wigner distribution synthesis technique [6]. SPWD images of trumpet and violin tones were used for warping and the effects of warping were evident in the resultant synthesised sounds. Warping of spectrograms can be accomplished by warping the magnitudes only and resynthesis using magnitude-only reconstruction techniques [7,8].

6. CONCLUSIONS

We have outlined a portable Java implementation of a warping tool for spectral transformation which uses the Java Advanced Image Processing API. The application offers an interactive method of spectral warping with ease of parameter specification and editing. Work is ongoing to include signal resynthesis from warped time-frequency spectra in the complete package [4,6,7,8]. Future work will include integration with other audio processing software tools [5]. Future development of the application would include extending these menus to include other features which would enhance the application.

7. REFERENCES

[1] Malcolm Slaney, Michele Covell and Bud Lassiter. "Automatic Audio Morphing", *Proceedings IEEE International Conference Acoustics, Speech and Signal Processing*, 2, pp. 1001-1004, 1996.

[2] Naotoshi Osaka. "Timbre interpolation of sounds using a sinusoidal model", *Proceedings ICMC* Sep. 1995, BANFF, Canada, pp 408-411.

[3] George Wolberg. *Digital Image Warping*, IEEE Press, 1992.

[4] Gloria Fay Boudreaux-Bartels. "Time-Varying Filtering and Signal Estimation Using Wigner Distribution Synthesis Techniques", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-34, No. 3, June 1986.

[5] Victor Lazzarini. "Audio Signal Processing and Object-Oriented Systems". *Proceedings Dafx02*, Hamburg, 2002.

[6] T. Lysaght and J. Timoney, ``Timbre morphing using the modal distribution", *COST-G6 Conference on Digital Audio Effects (DAFx02)*, University of Federal Armed Forces, Hamburg, Germany, pp. 191-194, September 26-28, 2002

[7] Puckette, M. "Phase-locked Vocoder." *Proceedings, IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, 1995

[8] Daniel W. Griffin, Jae S. Lim, "Signal Estimation from Modified Short-Time Fourier Transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236-243, 1984.