

Video Integration of PAM-VIEW Visualization Results

Sven Kuschfeldt^{†‡}, Michael Holzner[†],
Thomas Ertl[‡]

[†]BMW AG

[‡]Computer Graphics Group of the University of Erlangen

Abstract

A very efficient way to analyze computer simulations is to integrate the visualization of the results into video sequences. However, most postprocessors available today in the area of car crash simulations do not support the direct assembly of images into video. We have developed a software tool that allows for the annotation of images with text and graphics in connection with the PAM-VIEW¹ postprocessing system. It also supports the manipulation of the images and the archiving of image sequences as video clips. This is achieved using digital image capture, image compression and image manipulation facilities. The design and implementation of this software environment is described in this paper.

1 Introduction

At BMW, the PAM-VIEW postprocessing system is used to analyze the results of car crash simulations. PAM-VIEW is capable to display animations of the visualization of these simulations. The integration of the images together with synchronized time varying text had been achieved using a very limited annotation facility within a PAM-VIEW session by processing each frame individually. Additional annotations like explanatory graphics were inserted into the images in the following manner:

- an image to be modified was stored on hard disk using a special screenshot-tool within a PAM-VIEW session,
- the image was manipulated using special editing software and stored to optical disk thereafter.

After these image manipulations, the images stored on optical disk used to be assembled into a video presentation as follows:

- an image sequence was created using special software that arranges the display order of images,
- the image sequence was retrieved from optical disk, displayed on a television screen and recorded to VHS-videotape.

This method turned out to be very inefficient, considering the time required for the preparation of an image sequence. In order to overcome these disadvantages it is necessary to store PAM-VIEW image sequences digitally to hard disk and to provide convenient manipulation operations for digital image tracks. The manipulated images should then be used as input for

¹PAM-VIEW Version 1.2.1.

the creation of digital video clips, which finally can be archived to optical disk or directly recorded to VHS-video-tape. Based on these ideas we have developed a special system for the video generation of PAM-VIEW visualizations. This paper is based on previously published results [4] and describes the structure of this system as follows:

- Section 2 discusses digital capture of image sequences within a PAM-VIEW session.
- Section 3 shows how image sequences can be annotated automatically with static text and graphics.
- Section 4 describes the annotation of image sequences with synchronized time-varying text and diagrams.
- Section 5 explains the blending of several PAM-VIEW image sequences.
- Section 6 outlines the creation of digital image tracks from several image sequences and their archiving as digital videos on hard disk or as images on optical disk, respectively.
- Finally, in section 7, we present our conclusions and suggestions for further investigations.

2 Storage of PAM-VIEW animations as digital image sequences

PAM-VIEW processes the results of the crash simulation and displays an animation sequence representing the calculated crash states. After each state is displayed, a UNIX shell procedure can be called automatically, supplying the number of the state as argument.

We developed a procedure that can be used when running PAM-VIEW on Silicon Graphics workstations. At the beginning of the animation, while state 0 is displayed, this routine allows the user to select a portion of the screen for recording, and allows him to choose proper names for the resulting image files. During the animation, the selected screen area is saved into the given image files, and numbered according to the recorded state.

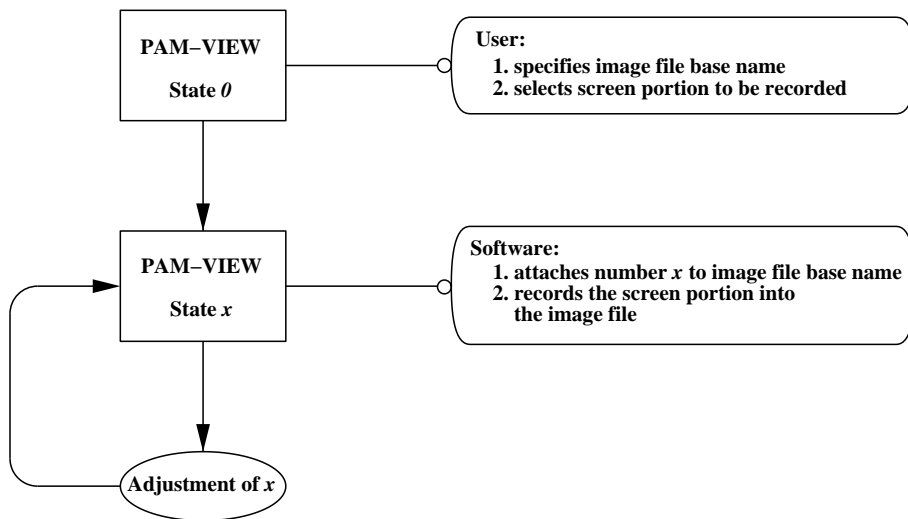


Figure 1: Digital storage of images within a PAM-VIEW session

The functionality shown in Figure 1 is achieved by implementing calls to following programs in the shell procedure:

- a **Motif**-based user interface [3] that allows the user to specify image file names,
- a C-program that uses the Silicon Graphics *Graphics Library* (GL) [2] to select a screen area to be recorded,
- a UNIX shell script that uses **awk**-routines [1] to number the image files according to frame number,
- a C-Programm that calls GL-routines in order to dump the contents of the selected screen area into the named and numbered image files.

After the animation is complete, a number of image files have been automatically generated and stored on hard disk. These files contain digitized images originating from PAM-VIEW and are stored in RGB format with one byte for the red, green and blue (RGB) components of each pixel of the image.

3 Annotation of PAM-VIEW image sequences with static text and graphics

Often it is very useful to annotate the raw PAM-VIEW image sequences with text (e.g. a heading) or with graphic primitives like arrows or bounding boxes. Our software tool allows for the following operations:

- Selection of the PAM-VIEW image sequence to be annotated using a **Motif**-based interface.
- Annotation of one image of the sequence taking advantage of the public domain color image editing tool **xpaint**.
- Automatic insertion of the annotation into all images of the image sequence using several C++ programs which call routines of the Silicon Graphics *Image Vision Library* (IL) [5].

The colors that can be used for the annotation are black, white, red, green, blue, magenta, cyan and yellow. All these colors are composed of red, green and blue components with either the maximum value (255) or the minimum value (0). For instance, red is represented by (255, 0, 0), blue by (0, 0, 255), and yellow by (255, 255, 0). The annotation is performed in several steps:

1. One image (denoted *image 1*) of the sequence is selected to be annotated.
2. The pixel values that represent the colors specified above are slightly changed (i.e., (255, 0, 0) is changed to (254, 0, 0)), if they occur within *image 1*. The differences between the original image and the modified image (denoted *image 2*) are practically invisible.
3. A copy of *image 2* is made and stored into *image 3*.
4. *Image 2* is annotated with the above mentioned colors using **xpaint** and stored into *image 4*.
5. When the annotation is complete, a pixelwise comparison of *image 3* and *image 4* is performed and a new image (denoted *image 5*) is created. If the color values of the pixel of *image 3* and the corresponding pixel of *image 4* are equal, the corresponding pixel value of *image 5* is set to the background color of *image 1*. Otherwise, it is set to the pixel value of *image 4*. As a result, *image 5* shows only the annotations made with **xpaint**.

6. All images of the image sequence (denoted *source image(1)*, *source image(2)*,...*source image(n)*) are processed frame by frame pixelwise examining *image 5*. A sequence of new images (denoted *new image(1)*, *new image(2)*,...*new image(n)*) is created and numbered accordingly. If a pixel of *image 5* has the background color, the pixel value of *new image(m)* is set to the value of the corresponding pixel of *source image(m)*. Otherwise, the pixel value is set to the pixel value of *image 5*.

The algorithm described above is outlined in Figure 2.

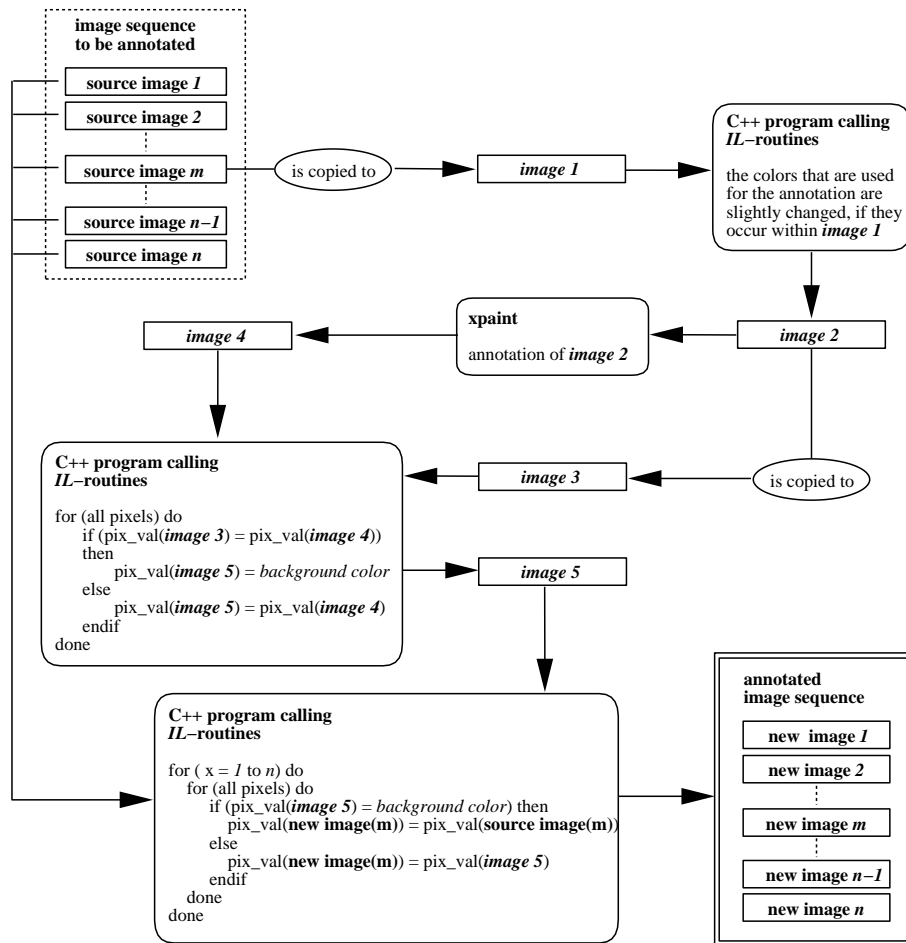


Figure 2: Static annotation of image sequences

The time required for the image manipulations that are performed when the annotation of *image 2* within *xpaint* is done, is mainly I/O dominated. On a Silicon Graphics Indy workstation (134 MHz Processor, 64Mbyte main memory size) the processing of one image takes about 2 seconds.

4 Annotation of image sequences with synchronized time-varying text and diagrams

PAM-VIEW allows the user to plot time history curves and to print other information (such as energies and global velocities) for a selected subset of nodes or elements of the FE model

for each calculated state. Since it is very useful to analyze such curves and the FE mesh simultaneously, we decided to implement the capability of annotating the image sequences with both curve animation and text representing the corresponding time step.

One possible way to reach this goal is to load the time history data provided by PAM-CRASH into a special program. This program then draws an animated curve onto a diagram, and/or displays text that shows the time state of the animation according to the displayed crash state. Finally, it inserts the graphics into the image sequence. In order to do this in a flexible way, the user must have the possibility to adjust size and position of the items to be inserted (see Figure 3). Then, each time a new state is reached, the appropriate diagram and/or text is copied into the image files, which are numbered according to the state. Thereafter, the target region of each image of the selected PAM-VIEW image sequence is pixelwise blended with the corresponding graphics image. For this "dynamic" annotation of image sequences we implemented the following programs:

- a Motif-based program for selecting the PAM-VIEW image sequence to be annotated,
- a C program containing GL routines to paste an image of the PAM-VIEW image sequence onto the screen,
- a FORTRAN program using GL routines that draws animated curves and text showing the different time states into two windows on the screen and allows the user to adjust size and position of these graphics windows,
- a C program containing GL routines to store the material to be inserted at each selected state into numbered image files,
- a C++ program containing IL routines that reads this graphics image files, performs the blending with the PAM-VIEW images using special pixel operations, and writes the results to new numbered image files.

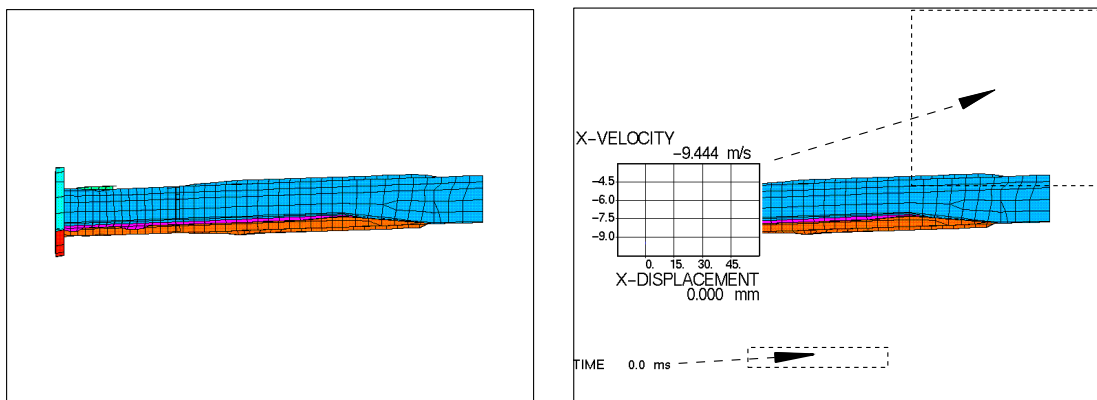


Figure 3: Adjustment of position and size of a diagram and a text to be inserted (right) into one image of the raw PAM-VIEW image sequence (left)

The synchronization of the blending process is managed by controlling the number attached to each image file. The numbers of both graphics image file and PAM-VIEW image file must be equal to guarantee that they describe the same state of the simulation. The annotation process begins by determining the regions of the PAM-VIEW images which are to be blended with the inserted graphics. This is followed by pixelwise examination of the graphics image. If the referenced pixel of the graphics image has the background color, the corresponding pixel of the new image is replaced by the value of the pixel of the PAM-VIEW image at this

position. Otherwise, the pixel value of the graphics image is used. The pixels of the PAM-VIEW image that are outside of the region to be annotated are copied to the corresponding positions in the new image. Now, the image shows the inserted graphics and the original PAM-VIEW image like layers that lie on top of each other (see Figure 4).

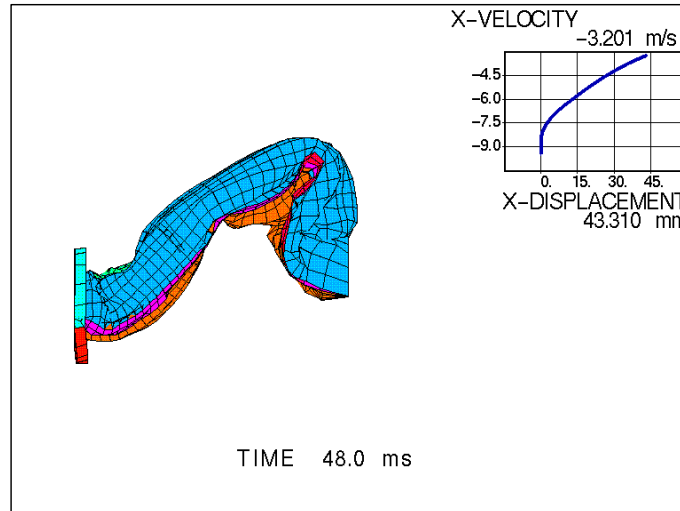


Figure 4: One image of the annotated PAM-VIEW image sequence

5 Blending of several image sequences

In some cases, it would be useful to display multiple PAM-VIEW image sequences simultaneously, for instance to compare different calculation variants of the same simulation model. In order to mix image sequences that show different cross sections, a similar scheme as in the previous section is used. The images referenced by the same attached numbers are processed pixelwise and the results of the blending are pixelwise written into new image files, which are numbered accordingly. If the pixel of the first image is of the background color, the pixel value of the corresponding pixel of the new image is replaced by the pixel value of the second image. Otherwise, the pixel value of the first image is used. The images of the sequences to be blended must be of the same size.

Another useful application is to blend several image sequences into predefined regions of a new image sequence. In order to achieve this, it is necessary to create a sequence (denoted *sequence 1*) of numbered background color images. Then, the first predefined region of each of these images is pixelwise mixed with the corresponding image of the first PAM-VIEW image sequence to be inserted and a new image sequence is created. If the referenced pixel of the PAM-VIEW image is of the background color, then, at the corresponding position in the new image, the pixel value is replaced by the pixel value of the image of *sequence 1*. Otherwise, the pixel value of the PAM-VIEW image is used. This procedure is repeated until all predefined regions of the new image sequence are filled with pixel data.

6 Archiving and compression of PAM-VIEW image tracks

After the capture and manipulation activities in processing PAM-VIEW image sequences it is necessary to arrange the manipulated sequences in right order and to archive the resulting image track either on hard disk or in analog format on optical disk. Another required func-

tionality is the capability to insert additional image sequences like opening credits into the image track. Our software tool meets these requirements using following programs:

- a Motif-based interface to select several image sequences and additional image files stored on hard disk and to arrange them into the desired order,
- a C++ program using IL routines that displays ordered image tracks on the screen and sends a signal to special soft- and hardware which stores this track on optical disk.

Image files on hard disk are usually stored digitally as values ranging from 0 to 255 for each of the red, green, blue and transparency components. This takes 4 bytes of storage per pixel, thus a 768x576 (PAL format) image would require nearly 1.3 megabytes of disk space. Obviously, the creation of image sequences produces an enormous amount of data that must be reduced during an image compression process. A lossless method of data compression is called run length encoding (RLE) and represents a line of pixels all of the same value by listing the value once, followed by the number of times it is repeated. When applied to PAM-VIEW images, this method is very efficient, because large areas of the images have the same color, namely the background color. There are several tools available that perform such compression. We use `rle` available on Silicon Graphics workstations. Considering the necessity of handling the image tracks on hard disk in a practical way we archive them as digital video clips. In order to get a compressed movie from an image sequence, we use SGI's `makemovie` that assembles a collection of images into a compressed video clip. A video clip of about 150 frames of an image size of 768x576 pixels takes from 5 to 30 Mbytes of storage space, depending on the amount of background color in the images, as opposed to the roughly 200 Mbytes which are required to store the 150 images without compression. The generated movie can be displayed on a Silicon Graphics Indy workstation at a rate of about 10 frames per second.

7 Conclusions

We presented a software environment for the analysis of car crash simulations which results in reduced costs in terms of processing time and in an improvement of presentation quality of PAM-VIEW visualization results.

One of the tools developed manages the storage of PAM-VIEW animations as digital image sequences on hard disk. The second one allows the annotation and blending of PAM-VIEW originated image sequences. Furthermore, it is possible to use this tool to archive image sequences in the form of ordered image tracks on optical disk as well as to create digital video clips in a convenient way. We customized existing tools and combined them with especially developed programs into a new software environment.

Further investigations are necessary for improving display performance in order to reach real time playback as well as in increasing compression ratios of the digital video clips.

References

- [1] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. *The awk Programming Language*. Addison-Wesley, 1988.
- [2] Patricia McLendon Creek. *Graphics Library Programming Guide*. Silicon Graphics, 1992.
- [3] Science+Computing GmbH. *Finesse User's Guide*. 1995.
- [4] Sven Kuschfeldt and Thomas Ertl. Digital video editing for the visualization of car crash simulation. In *Proceedings of the Dedicated Conference on Mechatronics, ISATA '95, 1995*.
- [5] Jackie Neider and Eleanor Basler. *Image Vision Library Programming Guide*. Silicon Graphics, 1993.