Indexing of Scenes of a Compressed Video Sequence

Karan M. Gupta
Department of Computer Science
Texas Tech University
gupta@cs.ttu.edu

Abstract

This report explains the implementation of a programming project to index the scenes of a compressed mpeg-1 movie. The programs have been written in C++ and compiled using g++ version 3.2.2 on a Linux machine. The project has been successfully ported to Solaris.

1 Introduction: The Objective

The objective of this project is to detect scene changes in a compressed movie and index them, with the output being presented as a mosaic of images.

The compressed movie sequence is a Mpeg-1 file. The project must return output in the form of YUV files of the final mosaic image. This image is finally viewed by converting the separate files to a ppm image.

2 Implementation

The implementation of this project has been broken up into several small programs which, when executed in proper order will generate a good mosaic based on the different scenes in the movie file. The project consists of eight programs, a shell script and a GNUPlot command file:

1. mosaic
2. filter
3. scale
4. Compute_Histo
5. SceneDetect
6. mpeg
7. FloorPlan
8. cyuv2ppm
9. P3_script
10. Plot_Histo

To run this project execute the P3_script at the command line. The programs are executed sequentially with the output of one being given as input to the next. The execution flow is maintained by the P3_script. The user is prompted for certain inputs during execution. The final mosaic file generated is myALBUM.ppm.

2.1 Developing a Mosaic

This program reads in a list of YUV files and generates a mosaic from them. The command to run this program is:

mosaic inputfile outputfile border_thickness

The inputfile given to the program is a preformatted text file (INPUT). Three files are generated as output from mosaic: outputfile.Y, outputfile.U and outputfile.V which can be converted into the final ppm file suitable for viewing. The border-thickness must be a multiple of 4. If it is not then a default border thickness of 8 is used in place of the one entered at the command line. In this implementation the P3_script asks for a border of 8 pixels to be applied to the mosaic. Processing is carried out in two stages, once for the Y component and then simultaneously for the U and V components. The mosaic program parses the INPUT file. The first line of INPUT gives values for M N W H, where M and N are the number

of rows and columns for the mosaic and W and H are the dimensions of Y components of the input images. After reading the first line from INPUT the program parses through the rest of the INPUT file, which contains names for the images to be used to create the mosaic. Four nested for-loops in the order: N H M W help to fill up the mosaic with the input images. A counter keeps track of which image the program must currently process. In the innermost loop, the program reads a pixel from the current input image and adds it to the mosaic image. After reading through the width of one image, the same procedure is repeated for the next image in the same row of the mosaic. This way, the program creates the mosaic row-by-row. A border is added to all the sides of the mosaic and in between the images.

After generating the YUV files for the mosaic, the program executes the cyuv2ppm program to generate a ppm image from the three components. The width and height of the new image are automatically calculated based on the borders added. After generating the ppm file, the program exits.

2.2 Filter Input Images

This program applies a filter onto an input image and generates as output the filtered image. The command to run this program is:

filter inputfile width height filter_number

The inputfile represents the following three files: inputfile.Y, inputfile.U, inputfile.V, which must be present in the same directory as this program. The outputfiles generated are: filteredinputfile.Y, filteredinputfile.U, and filteredinputfile.V which will be generated in the same directory. The width and height are of the Y component of the inputfile. Two Barlett filters are available in this program, one for a reduction factor of 2 and the other for a reduction factor of 3.

Either one of the two may be selected by passing the corresponding number as the argument to the program at the command line. The filter is applied to the three components of the inputfile separately. The file to be processed is read into a 2D-array the dimensions of which are equal to the width and height entered. Another array is also maintained which will hold the filtered image. To filter, a window, the size of the filter, containing the chosen filter's coefficients, is passed over the image. The numbers in the filter are multiplied with the corresponding coefficients of the image under the window. The results of these multiplications is accumulated in a sum variable. This is then divided by the normalization factor for the chosen filter. This result then replaces the centremost element in the image, under the window. For processing the U and V components, the width and height is halved and the same procedure is carried out, as it is for the Y component. As each component is processed completely, it is written out onto the output file.

2.3 Scale Input Images

This program scales an input image and generates as output the scaled image. The command to run this program is:

scale inputfile outputfile width height scalerate

The inputfile represents the following three files: inputfile.Y, inputfile.U, inputfile.V, which must be present in the same directory as this program. The outputfiles are outputfile.Y, outputfile.U, and outputfile.V which will be generated in the same directory. The width and height are of the Y component of the image.

Scaling for the each component is carried out separately. Each component image is copied into a 2D-array as it is processed. The dimensions of this array are based on the width and height entered. To scale a component, after storing the image into the array, the program writes it into the output file, skipping rows and columns, as decided by the scalerate. Therefore, to scale a component by 2, the program skips alternate rows and columns. For processing the U and V components the width and height is divided by 2 and the same procedure is carried out. For optimum results, the scalerate should be a multiple of 2. If the scalerate entered is `0` the program exits without doing anything.

2.4 Compute Histograms

This program generates the mean-absolute-difference of the histograms for U and V images

from a DCDUMP of every frame of the movie. The command to run this program is:

Compute_Histo dumpfile width height numoframes

The width and height are of the frames being processed. This program outputs two files: Histo_out_U and Histo_out_V. The dumpfile has been generated by the mpeg program. It contains a formatted list of the color value for the U and V component of every macroblock of every frame in the movie.

Based on the width and height of the frame the number of macroblocks per frame is calculated. The program maintains four arrays of 256 elements each, one element for each color, one pair of arrays for each U and V component. The program parses the formatted dumpfile one line at a time. It generates the histogram for a frame and stores it in an array. Then it generates the histogram for the next frame and stores that in another array. After that the histograms for both the frames are compared and a mean-absolute-difference is calculated. This value is output into a Histo_out file for the current frame. These operations are executed simultaneously for both U and V components. At the end of the process, the Histo_out files contain a list of numbers, the length of which is numoffframes-1. Every line has a value which is its MAD with the previous frame. These values will be used by SceneDetect to decide if a scene change has occurred.

2.5 Scene Detection

This program reads the Histo_out_U and Histo_out_V files and decides whether there has been a scene change or not. The command for executing this program is:

SceneDetect Uthreshold Vthreshold numoffframes

This program generates a text file: Index. This file contains frame numbers for one frame from each scene. The YUV files of the frame numbers specified in this file are used to create the INPUT file for mosaic. SceneDetect reads from the two Histo_out files and compares the MAD value for each frame with the Uthreshold and Vthreshold. If the MAD of either a U or a V frame is greater than its threshold, then a scene change is said to have occurred. Then the program calculates the number for a frame which is in the middle of the previous scene and sends that number as output to Index. Lastly, it outputs the total number of scenes detected in the movie.

2.6 Floor Planning

This program takes in a number of parameters and outputs an optimum plan for the mosaic, i.e the number of rows and columns of images, as well as if scaling is required or not. The command for executing this program is:

FloorPlan numofscenes width height screenW screenH scaleFactorList

Width and height are the dimensions of the video frames. ScreenW and screenH are the required width and height of the mosaic, respectively. This program outputs suitable values for M and N, as well as if any repeats will be required or not. This program has not been modified in any way from its original version.

The FloorPlan program also produces a scaleFactorList. However, this file is not being used in this implementation.

2.7 kmgHeader.h

This is a continuously developing custom header file which is being used in this project. It is being developed to incorporate some commonly used methods to speed up the coding process. Currently, it contains some simple functions to improve the feedback that a program can give to the user. At present it has two functions:
An overloaded function for drawing decorative lines.
A function for giving useful feedback to the user upon exit.

2.8 P3 Script

This script is responsible for proper sequential execution of all the programs. To run the script type at the command line:

P3_script video width height

When typing the video name, do not use the .mpeg extension. The width and height are the frame size of

the video. There is some error checking also in the script to maintain proper execution as far as possible.

This script uses the mpeg program in decoder mode to generate the YUV files for every frame in the video and puts these files in a directory of the same name as the video. It also computes the dc_dump by using mpeg program again, however this time in encoder mode. This dc_dump is passed to the Compute_Histo program. After the histograms have been generated, the script executes GNUPlot using the Plot_Histo command file to plot the histograms for the U and V images. The user can use this plot to decide on suitable values for Uthreshold and Vthreshold which are then passed as arguments to SceneDetect. SceneDetect generates a file called Index which contains a list of numbers: one frame number for every scene. The user is also presented with the total number of scenes that have been detected. The FloorPlan program uses this information to generate a suitable plan for the mosaic, i.e. the optimum number of rows and columns that the mosaic should have. It also tells the user if scaling of the input is required or not. The user is then prompted to enter a scaleFactor. If required, scaling is carried out by the scale program on the components of every frame specified in the Index, based on the scaleFactor. Then the script creates the file INPUT based on frame numbers in Index. The script now executes mosaic with INPUT being given as an argument to it. The script tells mosaic to add a border of 8 pixels to the image. The mosaic program generates the YUV files and also executes cyuv2ppm to generate the final ppm image of the mosaic. Finally the script cleans up all the temporary files and directories that were created during execution and exits.

Considerable changes were made to the script to accomodate this implementation. However, the original flow of the script has been kept the same. Instead of using a Compute_DC program, the mpeg program itself is used to generate the dc_dump file.

2.9 The Makefile

A makefile has been supplied with the project for easy recompilation. It has targets by the name of each program for compiling each program individually and also has the `all` target to compile all programs in one go. After compilation, the generated binary file is moved into `../bin`. The makefile assumes that the cpp files are in a `cpp/` directory and that there is a `bin/` directory at the same level which contains the executable files.

3 Testing

All the programs work as desired individually, and the project runs as desired, with the script. The mosaic, filter and scale programs were tested individually on lena (512 x 512). Only after every program worked as it should was it included in the script. The project has been tested using the following movies:

1. wg_cs_9.mpg: 7 scenes detected with UThreshold = 0.33 and VThreshold = 0.31. A 3x3 mosaic is generated with 2 repeats. 9 scenes detected with UThreshold = 0.33 and VThreshold = 0.28. A 3x3 mosaic is generated with no repeats.

2. son.mpg: 3 scenes detected with UThreshold = 0.6 and VThreshold = 0.6. Only first 171 frames were used. A 2x2 mosaic is generated with 1 repeat.

3. 6son.mpg: 6 scenes detected with UThreshold = 0.8 and VThreshold = 0.5. A 3x2 mosaic generated with no repeats. 5 scenes detected with UThreshold = 0.8 and VThreshold = 0.8. A 3x2 mosaic generated with 1 repeat.

The implementation works reasonably well for the above movies. Some points to note about this implementation:

- The filter program gives a crayonish touch to the image, therefore, filtering is not included in the process at this time.

- The scaleFactorlist is not being used. Instead the user is prompted to enter the value for the scalerate. The scaling will work for all whole number values, however, if an odd number is given, the resulting image's dimensions will have to be floored to whole numbers before conversion to ppm.

- The script has been modifed from its original version to accomodate the parameters that certain programs take. No changes have been made to the flow of execution of the script.
- No modification were made to the mpeg software.
- The mpeg program with the -4 option is being used to generate the DC_DUMP.
- The scalerate may be calculated automatically in the script. We know the required dimensions of the mosaic: screenW and screenH. We also know the width and height of each unscaled image. Therefore if the mosaic was created without scaling, the new width and the new height would be W*M and H*N respectively. Therefore, dividing these new values by screenW and screenH respectively we can get a scalerate.

I am still unclear about the filtering process. The filter is applied as specified, but the resulting image has a pastel shade to it.