Machine Problem 1: Video Recorder and Player

CS414 Spring 2011: Multimedia Systems
Instructor: Klara Nahrstedt

Posted: Jan 28, 2011
Due: 11:59pm Feb 16, 2011

Introduction

While you are chatting with your parents through facetime, watching the fun stuff from YouTube, listening to the new album from Pandora, have you ever wondered how these voice, music, and motion images are transformed to binary bits, transmitted from one place to another place, and finally played on your mobile phone or PC? This video conferencing project will help you figure out all details.

In this first MP, we will focus on the recording and playback of video/audio contents. You will learn how to access the raw images captured by the webcam (fortunately, you do not need to worry about how those raw images are generated inside webcam), how to transform and compress those raw images, how to display sequential images on screen as a video, how to sample sound waves to bytes, and how to play bytes to sound waves. Besides, you will also have some basic understandings of audio/video synchronization and what "MUX" and "DEMUX" really mean.

You are required to work on Linux for this machine problem. You can use the Linux workstations in the EWS lab room SC216 and SC220. A group directory with sufficient disk space will be set up for everyone. You can use your own machine as well. The recommended Linux version to install is Ubuntu 10.04 or newer. C/C++ is the recommended language for your program. However, if you insist using Java or other language, you have to deal with driver problems on your own because this MP is very related with audio/video devices for your MP. Each group can borrow two Logitech QuickCam Orbit MP webcams from TSG. The Logitech webcam can also be used as the microphone for audio recording. You need to prepare your own headphone/speaker to test the sound playback.

Problem Description

Implement two programs named as *recorder* and *player*. *recorder* should be able to capture the video frames from the webcam, collect the audio data from the microphone, and save both video and audio data in one file. *player* should be able to play the media files pre-saved by *recorder*. Here are the detail feature requirements for two programs:

Required Features (80 points)

1. *Video Recording (10 points)*: *recorder* can access the webcam through V4L2 interface and save the captured image frames. The image resolution should be no smaller than 320 $\times$240 and the capturing speed should be no less than 15 frames per second.

2. *Video Window (10 points)*: *recorder* opens a video window on screen showing the video that is currently recorded.

3. *Video Compression (10 points)*: *recorder* should compress each video frame with JPEG or more advanced video coding techniques.

4. *Audio Recording (10 points)*: *recorder* captures the raw audio signals from the microphone (you can use the microphone embedded in the Logitech Webcam or your own individual microphone).

5. *A/V Muxer (10 points)*: *recorder* captures both video and audio data concurrently and saves in one file. You can define you own video file format and mux scheme. Saving audio and video to different files only gets partial points.

6. *Video Play (10 points)*: *player* can read the media file generated by *recorder* and display video frames on the screen. The frame rate of video playback should be calculated and displayed during or after playback.

7. *Audio Play (10 points)*: *player* can read the media file generated by *recorder* and play the audio data out to the speaker. The sound quality should not be distorted.

8. *A/V Sync (10 points)*: *player* should synchronize the playback of video and audio streams all the time.

Optional Features (40 points)

9. *GUI (5 points)*: design any graphic user interface can get 5 points.

10. *Format (10 points)*: *recorder* saves the recorded audio and video data using a well-known media file format (e.g., avi), so that the media file can be played by standard media players (e.g., mplayer). You also get full points for feature 5 (A/V Muxer) if you implement this one.

11. *Pan/Tilt (10 points)*: the reason we use Logitecth Orbit webcam is that it can pan and tilt. Add control to your *recorder* to enable paning and tilting the camera while recording the video.

12. *Audio Compression (10 points)*: *recorder* can compress the raw audio data with MP3 or AAC standard.

13. *Adv. Video Compression (5 points)*: *recorder* can compress the video frames with MPEG-4 or H.264 standard (you can choose any profile). You also get full points for feature 3 if you implement this one.

Examples

If you design GUI for your programs, the user interface should be easy to understand. Add illustrations in your documents if anything is confusing. If you just design a command line program, here is an example for you to follow. Make sure to write usage explanation in your documents if your implementation is different from the example below.

Both *recorder* and *player* take only one parameter to indicate the file name that the program writes to or reads from.

recorder FILE_NAME

means the video and audio data will be recorded to FILE_NAME. A video window pops out and shows the video currently recorded. Press CTRL+C to stop recording and quit the program.

player FILE_NAME

plays the pre-recorded file FILE_NAME. A video window pops out to play the video and the audio is played in the speaker. The program quits when the media file reaches the end and the playback frame rate is printed on screen like this:

```
Playback Frame Rate: 15.4 fps
```

Submission

An empty sample solution is provided. Follow all naming rules and directory organization to prepare your solution. Pack all source codes and documents into a zip file or tar ball and submit through Compass. Do not submit your solutions through email unless there are technical problems with the Compass system. Late submissions are not accepted and will get 0 point.

Source Code

You should only submit your source codes (.h, .c or .cpp files), Makefile, and any open source libraries you use in your solution. Do not include any pre-compiled obj files (.o), binary execution, or pre-recorded media files in the directory.

All programs should be built by Makefile. Google for tutorials if you have never used it before, or you can modify the Makefile in the sample solution for your program.

Documentation

You need to submit two documents: user manual and development manual. The user manual should inlcude all instructions on how to compile and install your source code, and how to run your program and test all features. If you have designed any GUI, it is better to attach some screen shots to explain. The development manual should have the implementation details of all features. The implementation details include program flow, key data structures, media file formats, important algorithms, and so on.

Evaluation

Your solution is evaluated based on how many features you have implemented. In order to get full score (100 points) of this MP, you need to implement all 80-point required features and any 20-point optional features. If you get more than 100 points, you will get bonus points for your final score.

For each feature, you get 20% of the allocated points if the feature is implemented and well documented; 20% if the source codes can compile and run; 20% if the feature is partially implemented; 20% if the feature is fully implemented; 20% if the program runs perfectly (no error, no crash).

Comments

- This MP may look easy but actually can take much more time to finish than you think. So start early!
- Always make sure your Logitech webcam functions correctly and the system driver works before testing your own program. You can test it by running `luvcview` (included in reference materials) or `mplayer` (simply run: `mplayer tv://`).
- The more you read (reference materials, web tutorials, open source codes), the less you write (your own implementation code).
- Collaborate with your teammates. The project is designed for three students.
- Keywords for required features: V4L2, UVC, OSS/ALSA, SDL, libjpeg, PTS (Presentation Time Stamp).
- Keywords for optional features: GTK, AVILIB, uvcdynctrl, lame, ffmpeg.
- Come to the MP help session on Feb 9th.
- Keep in mind that your codes for this MP will be reused for MP2 and MP3. If you plan to hack some open source software, make sure you really understand how it works. Otherwise, you will have to re-implement this MP later.
- Follow newsgroup. Your problem can have been solved there.