The MPJEG Tools

The MJPEG Tools suite (http://mjpeg.sourceforge.net/) is a group of utilities used for video manipulation. When it comes to video processing, many free software users don't look beyond the well known mainstream utilities, like transcode (http://www.theorie.physik.uni-goettingen.de/~ostreich/transcode/) and mplayer (http://www.mplayerhq.hu/homepage/design7/news.html). When these utilities aren't producing the desired output, or when the optimum solution can't be found amongst their hundreds of different options, it's is useful to know of another toolset which might be able to help. The MJPEG Tools suite is just such a toolset. Mature, feature rich, and of extremely high quality, the MJPEG Tools provide video manipulation facilities which can be used either as a complete end to end solution, or as a toolbox from which the exact tool for a given job can be selected. Whether you need assistance with capturing, improving or encoding video, the MJPEG Tools can help.

A Little MJPEG Tools History

The original intent of the MJPEG Tools was to provide a package which would enable Linux users to capture and playback video through a PCI card based around the Zoran ZR36067 MJPEG chip. Wrapped in the standard Audio/Video Interleaved (AVI) container format, MJPEG – properly known as "Motion JPEG" - is essentially a sequence of JPEG still images which when played back fast enough show as a movie. The main reason most people don't come across MJPEG as a day to day video format is that it produces enormous video files – it can be several megabytes a second – so it's a terrible format for sharing video. The flip side of the coin, however, is that MJPEG is an ideal format for video editing. There are no key frames, and no part of the video is built from adjacent frames like most compressed formats. You can cut, copy, splice and generally edit sections of an MJPEG stream quickly and accurately on an individual frame basis.

Zoran based capture cards, like the Pinnacle DC10+ and the Iomega Buz, capture and compress the JPEG frames in realtime, sending MJPEG video straight into the computer. The MJPEG Tools project started out with a couple of useful utilities to allow you to use these cards under Linux. Once PCs became powerful enough to do the capture and compression in realtime, the tools were expanded to allow software encoding from less capable hardware such as simple BT8x8 based TV capture cards. From there the post processing capabilities of the tool suite started to grow, and it's this part of the package which is of most interest to many people today. Figure 1 shows where the MJPEG Tools can fit into a video processing workflow.

Capturing Video

Zoran based capture cards fell into obsolescence some time ago, and although they are still fairly easy to find via eBay, many people will probably now choose a digital video (DV) based solution. For those who do have one of these Zoran based cards, or who have another analogue capture device, the capturing facilities of the MJPEG Tools are still extremely useable, so we'll start with those.

Before we do so, however, it should be pointed out that although the capture itself happens in MJPEG format, the majority of the tools actually use a more raw format called YUV4MPEG2. The MJPEG video stream is typically converted (on the fly) into a YUV4MPEG2 video stream, because that's the actual format most of the MJPEG Tools use.

The capture tool in the MJPEG Tools is called lavrec, and it can be used to capture directly from a Zoran based capture card or from a regular video capture card using software encoding. Either way, the result on the disk is an AVI file containing a MJPEG video stream. This video stream can be edited using another tool in the MJPEG Suite called glav, which is a simple GUI based video editor – see Figure 2. glav allows you to play the MJPEG video forwards and backwards at various speeds, to select sections and cut, copy and paste them. It ultimately creates what is known as an edit list which is a small text file which describes the edited video sequence. An edit list file can be used in place of an MJPEG AVI file as input to many of the tools in the MJPEG suite.
The final step in this part of the process is to convert the MJPEG stream (or edit list) into the YUV4MPEG2 format described previously. The tool to do this is called lav2yuv, which sends the YUV4MPEG2 stream to its standard output.
As indicated in Figure 1, for those with DV based equipment a similar tool chain (http://kino.schirmacher.de/) exists. This article won't go into details, but the chain basically consists of a DV frame grabber such as dvgrab, then using the kino video editor for removal of unwanted sections and the splicing in of new parts. The DV equivalent of the edit list file is an SMIL file, and it's this file that you give to one of the tools in the smilutils package (specifically smil2yuv) which emits it as a YUV4MPEG2 stream. From there the MJPEG Tools utilities can be used on the video.

Filters

What happens once lav2yuv (or smil2yuv) has converted the video from your VCR or camcorder into a YUV4MPEG2 stream depends on what you want to end up with. The end goal will often be a file you can burn onto disc for playing on your DVD player, or maybe send to friends and relatives to play on a computer. Before you do that you might consider employing a few filters to tidy the video up a bit, and this is the area where the MJPEG Tools really shine.
The design of the MJPEG suite follows the UNIX philosophy of "one tool to do one job", and each tool uses its standard input and standard output streams. This means it is possible to build up a filter chain using a traditional UNIX pipeline. Let's look at some of the options.
The first thing you might want to do is denoise the video. If the video came from a nice clear source this may not be necessary, but if the video came from an old VHS tape or other noisy source, this step removes, or at least reduces, all those flickers and specks. The MJPEG yuvdenoise utility can take a few parameters for fine tuning thresholds and such, but for the most part it adapts dynamically to the content of the video it finds. It is typically run without parameters, like this:

```
lav2yuv myvideo.eli | yuvdenoise | …
```

with the lav2yuv part being replaced by smil2yuv if the source came via DV.
The next step is quite often the software scaler, which changes the size of the video frame. The frame size would normally be whatever the native size of the capture card is, and that varies between devices. You will likely need to rescale to match the requirements of your destination medium. The MJPEG Tools yuvscaler can scale up or down (i.e. bigger or smaller), and can use any of a number of algorithms to do the work. If required you can choose the exact size of output you want, but most people would probably pick one of the standard modes which are preconfigured for VCD, SVCD and DVD.

If our example pipeline were heading towards a DVD output, a yuvscaler command would be:

lav2yuv myvideo.eli | yuvdenoise | yuvscaler –O DVD | …

For a particularly noisy video source another filter is sometimes useful. This one is the yuvmedianfilter, which improves the image by removing low frequency noise. The (quite excellent) MJPEG Tools HOWTO (https://sourceforge.net/docman/display_doc.php?docid=3456&group_id=5776) has this to say on the subject:

“Using yuvmedianfilter's capability to only filter the chroma (-T) is moderately effective at reducing noise in dark scenes without softening the image during normal (brighter) scenes. Median filtering of the luma (-t) will produce a lower bitrate but can cause loss of detail (softening). Chroma only medianfiltering is less agressive and is a good choice to use in combination with yuvdenoise.”

This kind of language is rather typical of most video manipulation documentation, which is no doubt a great thing if you understand what the heck it’s on about, but a bit intimidating if you don’t. But if you keep track of the goal, rather than trying to understand the process too deeply, it’s not too hard to get improved results. The HOWTO information above, together with a read of the yuvmedianfilter man page, suggests that if our video has excessive noise in the dark scenes, adding this filter to the chain might help:

lav2yuv myvideo.eli | yuvdenoise | yuvscaler –O DVD | yuvmedianfilter –T 3 | …

Then again, it might not. Or maybe it will help if we put it in before the scaling step. The thing about these filters is that until you have experience with them the best way to find out if they help is just to try them on a 5 second clip and fiddle with the parameters a bit. You can see how things are looking by putting a yuvplay on the end of the chain, which sends the video to the screen:

lav2yuv myvideo.eli | yuvdenoise | yuvscaler –O DVD | yuvmedianfilter –T 3 | yuvplay

There are other filters which are useful in certain situations. For example, if you have a video source which has fuzzy edges (video captured from a VCR often has this) you should use a filter that gets rid of it. That fuzziness is a distraction which adds nothing, and actually takes a lot of effort to store and reproduce! Use a filter to crop it off or black it out.
There are lots of other filters which you can use to generate video from still images, change the audio track, and so on. Sometimes it’s hard to know when to stop

Encoding

Once you finally have a chain of filters which produces a video stream as you want it, you have to encode it to the format required by your output medium. The MJPEG Tools specialise in producing MPEG output destined for VCD or DVD players. There is a script in the MJPEG Tools package called lav2mpeg which will do a straightforward encoding of your MJPEG file to one of a set selection of output

formats. This script might occasionally do what you want, but best quality output will always be achieved using the individual tools so as to keep control of the steps.
Step one of the encoding process is normally to extract the sound from the original video file. For DVDs or VCDs the target is the MPEG-1 Layer 2 sound format. The MJPEG Suite provides a tool to extract the video's audio track to a WAV file, and another to encode that WAV into MP2 format. These tools are called lav2wav and mp2enc respectively and they are normally used with a command like:

```
lav2wav myvideo.eli | mp2enc –o myvideo.mp2
```

There are various switches to control the format of the output file, or you could use an alternative MPEG audio encoder like toolame (http://www.planckenergy.com) if you're more familiar with that. Note that the command above doesn't apply any filters – there's no point in denoising a video stream if you're only going to extract the audio from it!
Step two would be to encode the video. The video encoder in the MJPEG Tools is called mpeg2enc, and here lies a slight problem. Because of some legal uncertainties with software which uses the MPEG-2 codec, some Linux distributors play safe and choose not to include the mpeg2enc binary in their MJPEG Tools package. SUSE is one such distribution, so SUSE users would probably want to grab the MJPEG Tools package (http://packman.links2linux.org/?action=154) from packman (http://packman.links2linux.org/) which is complete; other distributions might have a similar alternative package, or there's always the source.
mpeg2enc encoding is where having a good, clean video stream from your selected filters makes a big difference. The better the video you put in, the better the encoded output will be. The basic command for mpeg2enc looks something like this:

```
lav2yuv myvideo.eli | <filter chain> | mpeg2enc –f 8 –o myvideo.m2v
```

The mpeg encoder has a number of predefined formats, of which number 8 is standard format DVD. There are other formats for VCD, SVCD and so on, and lots of other options for special requirements - see the man page.
The output from mpeg2enc is either MPEG-1 video or MPEG-2 video, without sound. The final step is to merge the sound file and the video file, the tool for which is called a multiplexer. The multiplexer in the MJPEG Tools suite is called mplex, and it might be used like this:

```
mplex -f 8 myvideo.mp2 myvideo.m2v -o myvideo.mpg
```

Again I use the preset format number 8 which produces a file suitable for burning onto DVD and watching with a regular DVD player. There are other presets and lots of options to fine tune the results.
A popular alternative to all this is to encode to DivX (http://www.divx.com/) (MPEG4) format. Although DivX is still almost completely incompatible with hardware DVD players, it is very popular for files destined to be played on computers. MJPEG Tools defers the creation of DivX video to other applications which are more suited to the task, such as mencoder, transcode or avifile. There is a demonstration script in the MJPEG Tools package called lav2avi that does a basic job of DivX encoding using mencoder, and which can be used as a basis for a more fine tuned version if required.

Conclusion

Video processing is one of those endeavours where having a large, well equipped toolkit is better than having a single program which tries to do everything. The subject is just so wide ranging, and can present so many diverse problems, that it's hard to predict what difficulty your project is likely to present you with next.

This is, therefore, an area where the UNIX philosophy of one tool for one job really shines, and the MJPEG Tools suite is an excellent implementation of that philosophy. Its video capture features are useful to people with the right hardware, its filters are useful to people with videos which need cleaning up or reformatting in some way, and its encoding tools are useful to people who need to get their videos into the required destination formats. It plays nicely with the other major tools in the area, and lends itself to the idea of just picking one tool from the suite if that is all that is required to improve an established workflow.

Video processing tasks are rarely easy, but most things are possible given the right tools, which is why the MJPEG Suite should definitely be part of your toolbox.

My thanks to Steven M. Schultz from the MJPEG users mailing list (http://sourceforge.net/mailarchive/forum.php?forum_id=2356) who provided technical assistance with this article.

<bio>