Using convmlv 1.7.1

Developing Magic Lantern footage with ease

by Sofus Rose

March 28, 2016

Contents

1 Introduction

Processing the output of Magic Lantern for maximum quality is a technical balancing act that most don't want to deal with. Even if you do, it's often tedious, *especially* on Linux. convmlv aims to make the development of MLV, RAW, and even DNG sequences as easy as possible without compromising quality, whilst exposing as much depth as possible for those who wish to tweak.

Yes, it's a command line program, but don't let that scare you! 'cd newFolder' lets you change your current folder to newFolder, and 'ls' lets you list all the files in your current folder; that's all you need to follow along here!

2 Installation

2.1 General info

convmlv comes in the form of a ".sh" script. Installation is a little hairy, but bear with me, and you'll be done in max 10 minutes!

Besides the script itself, several **dependencies** are required. How to best get these varies by system.

To begin, download **convmlv.sh** and **balance.py** to the same folder from `https://bitbucket.org/so-rose/convmlv`.

2.2 UNIX

On UNIX systems, everything is easy. There is certain setup that must be done, however. Open up a terminal, navigate to the folder containing convmlv and balance.py, and type each line:

```
chmod +x convmlv.sh balance.py #Let convmlv execute.
```

You can run the script now, but it won't work! This is because you must install the dependencies. First off, these are the package dependencies; to list all package dependencies, type this:

```
./convmlv.sh -K
```

Find and install the corresponding packages on your machine. Imagemagick must be installed with exr support! If you're on Debian, this task is very, very simple. Just type (or paste):

```
sudo apt-get install $(./convmlv.sh -K) #$Only for debian.
```

Next, you must install the Python dependencies. This is universally very easy, once you have pip or pip3 installed (try both). On Debian, simply type:

```
sudo pip3 install $(./convmlv.sh -Y) #$Python 3 version of pip.
```

Finally, you must install the manual dependencies made by the talented Magic Lantern community. You can list these by typing './convmlv.sh -N'.

The Easy Way: Download each item in the list, making sure you put it in the same folder as convmlv.sh. 'chmod +x' each file by itself, or just run:

```
chmod +x $(./convmlv.sh -N) #$Batch chmod
```

If you want to be able to type convmlv instead of ./convmlv.sh, you need to link it to your PATH. On Linux in general:

```
sudo ln -s $(pwd)/convmlv.sh /usr/local/bin/convmlv #$ Link for execution.
```

That's it! You're all ready to go!

2.3 Mac

Follow the above UNIX directions, using the Mac Terminal. Instead of the apt-get command, try homebrew and/or install the dependencies manually. If you can access the programs from the command line, then you're golden.

There are no other differences.

3 Output Options

The development of an mlv named "footage.mlv" in the current directory into a ProRes video is very, very simple:

```
convmlv -m footage.mlv
```

A folder named 'raw-conv' will be created, inside of which a folder named 'footage' will be created, inside of which you'll find your high-quality, ready to edit ProRes video! By no means, however, is this the limit of the script.

Image sequences, the standard in high-quality color grading and VFX, can be outputted either compressed or uncompressed as such:

```
convmlv -i footage.mlv
```

In the same 'footage' folder, you'll find a folder named exr-footage containing a sequence of OpenEXR files! Other formats are available; refer to './convmlv.sh -h'. If you want to compress the images, simply add -c, and the best lossless compression will be chosen automatically (for some formats, this might incur slowdowns):

```
convmlv -i -c footage.mlv
```

With these large files, slowdowns can often happen during heavy editing/grading. Sometimes you just want a preview as well. As such, you're able to generate 'proxies' in the form of either an mp4 (mode 1), a jpg sequence (mode 2) or both (mode 3). Specify the mode with the -p option as such:

```
convmlv -i -p3 -c footage.mlv #Will generate mp4/jpg proxy.
```

Finally, by default the proxies are 1/2 the scale of the original footage, so that editing them is faster. Specify the proxy scale with -s, in percentages, as such:

```
convmlv -i -s75% -p3 -c footage.mlv #75% proxy scale.
```

JPG proxies won't be generated without the -i option, but mp4 proxies **will** be generated without the -m option. Finally, also keep in mind that you can create both sequences and videos - just specify both -i and -m.

4 RAW Development

convmlv uses DCraw, a powerful RAW development tool, for many of its functions here. I'll go through them in detail here:

- Demosaicing, -d: RAW images aren't really images; it's literally what the sensor spits out. As such, it's neccessary to process this sensor data, which has pixels for R, G, and B laid out in what's called a Bayer grid, into a viewable image. Algorithms that do this are called demosaicing algorithms, and you have choices! Lower numbers are faster, higher numbers are higher quality. For example, to generate the best quality compressd images, specify:

  ```
  convmlv -i -c -d3 footage.mlv #High quality demosaicing.
  ```

- Four Color Mode, -r: When the VNG and AHD demosaicing modes get strange, this option tends to fix things.

- Noise Reduction, -n: RAW processing is a great point in the pipeline to do some wavelet denoising, if you wish, which reduces noise, but also detail. A setting of 200 can save images that might have been too noisy.

- Gamma, -g: By default, the output is in Linear color space to preserve maximum quality. If you wish, however, your output can be graded to a variety of curves, including sRGB, Adobe RGB, and BT.709.

- Bit Depth, -S: The quality of RAW comes from its high bit depth of 14, which is saved as 16 bits. If you only want 8 bits, however, just specify -S. You really do want 16 bits though...

5 White Balance

In convmlv, Auto White Balance is used by default. If you wish to use a different white balance mode, specify -w. For example, to use the camera white balance, specify -w1 (in my experience, it's often not very helpful):

```
convmlv -i -c -w1 footage.mlv #30 AWB samples.
```

If you do not wish to balance the image at all, specify -w2. This will make for a crappy image if you don't balance it yourself later!

When using AWB, what balance.py does is average the white balance of 15 frames, spread evenly throughout the footage, then apply the averaged balance to the entire shot. You can change the amount of samples that it will try to average the white balance using with the -A option, for speed vs. accuracy:

```
convmlv -i -c -A30 footage.mlv #30 AWB samples.
```

6 Features

6.1 Dual ISO

convmlv is capable of processing Dual ISO footage! Simply specify -u.

6.2 Badpixels

On some cameras, focus pixel issues are rampant. To fix it, simply specify -b! For Dual ISO footage, this is not always totally effective; the issue is being worked on!

6.3 Darkframe Subtraction

In reducing noise, you can do a very sneaky technique called darkframe subtraction. It's very simple:

- Dark Footage: With your lens cap on, using the same settings as your footage, take 5 seconds of RAW footage.
- Use Dark Footage: Specify -F, then the path to the dark footage, to automatically average the footage + reduce noise! Example:

```
convmlv -i -c -F./dark.mlv footage.mlv #30 AWB samples.
```

For a decent speedup when batch processing, you are able to **preaverage** your dark footage. Use mlv-dump to do this, making sure to save the output file with the .darkframe extension (I'll explain why in a moment). Example:

```
./mlv_dump -o 1600iso.darkframe -a dark.mlv
#./mlv_dump -o outputfile.darkframe -a darkmlv.mlv
```

To use the averaged file as an averaged file, make sure it has the .darkframe extension, and then just specify it in the -F path. In this way, you're able to build up a library of sorts at different ISOs! Example:

```
convmlv -i -c -F./1600iso.darkframe footage.mlv
```

6.4 Applying LUTs

Using convmlv, you're able to apply LUTs to images and movies! EXR sequences and image proxies don't work, and images in general are slow to apply, but otherwise it works fast and fine with the option -l, then the path to the LUT! Example:

```
convmlv -m -l./expensiveLUT.cube footage.mlv
```

6.5 Custom Paths

If you want to use a custom path for any of the manual dependencies, you can. This is the first section of options when you specify 'convmlv -h'. Additionally you can redefine the output directory, which defaults to raw-conv. The directory will then automatically be created for you. For example:

```
convmlv -i -c -d3 -o./differentoutdir footage.mlv
```

7 Misc Considerations

convmlv is multithreaded; you can redefine the number of processes in use from a default of 8 using -T followed by an integer.

Batch processing files can be done with a *. For example, to process all .MLV files in a directory:

```
convmlv -i -c *.MLV
```

If you have any feedback or feature requests, I'd be happy to hear it! Just reply to the thread at `http://www.magiclantern.fm/forum/index.php?topic=16799.0`, or PM me on the forums, or write a comment in the repository!