Lab-8: ImageMagick or Making Movies

Apr.-27, 2010

You all now have at least 2 images of reflectivity of southern Iceland and adjacent waters in the visible (Modis band-1) and near infrared (Modis band-2) for a 2-week period. We here will regard each of these as a single frame of a movie, not great, but you will get the idea and perhaps add frames to bring it to life with more frames and work together with other groups that, after some co-ordination, will have additional frames to offer for other 2-week periods.

A free, open-source, and powerful general utility image manipulation software is ImageMagick available with documentation at

http://www.imagemagick.org

which is already installed on our server. It runs in a command-line setting, provides interfaces to a variety of programming environments, and can be used within Windows, Linux, and Mac OS X operating systems (our server runs Mac OS X). Commands can be as simple as

```
convert image.jpg image.png
```

or as complex as

```
convert label.gif + matte\
\( + clone –shade 110x90 –normalize –negate + clone –compose Plus -composite \) \
\( -close 0 –shade 110x50 –normalize –cannel BG –fx = + channel –matte \) \
-delete = + swap –compose Multiply –composite button.gif
```

Here are the steps to make a movie, that perhaps all could be collected in a shell called movie.csh. Within this shell, do the following:

#1. Convert all postscript files (one per frame) generated by GMT to .png using the ImageMagick "convert" utility

```
convert InputFile1.ps OutputFile1.png
convert InputFile2.ps OutputFile2.png
…
```

#2. String these .png files together as an animated .gif file again using 'convert"

```
convert –set delay 6 –colorspace GRAY –colors 16 –dispose 1 –loop 0 –scale 100% Output*.png Movie.gif
```

You can think of "convert" as a filter that takes a single input file (as in #1) or a set of input files (as in #2) and converts them to an output. In #2 the set of input files are frames of a movie in alphabetical order, so the file A2010.png will come before T2009.png, so your file naming will determine which frame comes first.

For only 2 or 10 frames, manually entering the items needed for #1 is not a bad approach, however, once you deal with 20 or 50 or 500 frames, it is best to look for an automated way to do that. Can you perhaps think of a way to automate the task in #1? As a hint, my own approach starts with these line in my movie.csh:

```
set id = $argv[1]
ls –la $id*.ps | nawk 'BEGIN{print "#! /bin/csh"};{print "echo",$9;print "convert",$9,substr($9,1,14)".png"}'
>My-ps-2-png.csh
```

What am I doing or try to do? What will the next line in this code likely be?