Batch-oriented MPEG generation with GMV in background mode

Release 2.0

Sven H.M. Buijssen, Stefan Turek

Institute of Applied Mathematics, University of Dortmund
Vogelpothsweg 87, 44227 Dortmund, Germany
Email: gmvmpeg@featflow.de
Homepage: www.featflow.de

September 2001

Abstract

For automatisation purposes we developed a perl-script called `gmvmpeg` that simplifies the generation of MPEG movies with `GMV`, the "General Mesh Viewer" of the Los Alamos group. Particularly, the possibility to do this "in background" on (remote) computers might be of general interest.

What it does

There are numerous software packages for the visualization of numerical data. Within the FEATFLOW software, `GMV` (General Mesh Viewer) of Los Alamos National Laboratory belongs to our favourite tools. For an overview of the features of this program please see its documentation [Ort01].

Mainly two features of `GMV` are exploited by the script `gmvmpeg`: `GMV` comes with the possibility to save arbitrary settings into an attribute file. Thus, you only have to choose what to visualize (cutplanes, isosurfaces, isovolumes, particles etc.) and put this to disk. Next, we can use the batch mode of `GMV` together with this attribute file to visualize all files of our sequence. The images we get are finally passed to an MPEG encoder to create a video.

How to use

Once you succeeded in creating a coarse mesh with DEVISORGRID, prescribing boundary conditions in FEATFLOW and running FEATFLOW on your computer, you end up with a bunch of `GMV` files. Especially if you have an instationary configuration. Most likely, you want to visualize your calculations by creating an MPEG movie. With `gmvmpeg` all you need to do in order to generate an MPEG movie out of a sequence of files containing data from a numerical simulation is the following:

1. Start `GMV`
2. Load a (more or less) arbitrary file from your sequence
3. Make a decision on the subset of data to be displayed.[1]

[1] The manuals [Ack98], [AT99] and especially the complete `GMV`manual [Ort01] will help you withal.

4. Save your settings into a so-called attribute file.
5. Finally run the perl-script `gmvmpeg`with appropriate command lines options (see below).

Installation and configuration

The installation process is as simple as copying this script (which can be obtained from the FEATFLOW homepage) to a directory you like. But you probably have to adjust a few variables within it. The script needs to know the location of the following programs:

- `GMV` for the visualization process; can be downloaded from the `GMV` homepage (see below).

  `GMV` 2.7 or above is required.
- `sgitopnm` and `ppmtoyuvsplit` from the `netpbm` package for converting the `GMV` images into a format the MPEG encoder can handle
- `mpeg` for the encoding process (see below).

Additional, if you want on-the-fly decompression of compressed `GMV`files during MPEG generation, you have to specify the program locations of the following supported compression utilities: `gzip`, `compress` and `bzip2`.

Command line options

`gmvmpeg` has the following command line options to control the noninteractive generation of MPEG movies.

attribute file: **-a filename**

- Path of the `GMV` attribute file to be used.
- Defaults to "default.attr"

prefix of input files: **-i prefix**

- `gmvmpeg` assumes the following structure for the input files containing the data to be visualized:

  <prefix>.<number>.gmv,

  where <number> is supposed to have no additional prefix zeros.

  If you do not like this naming pattern, you can easily adapt the script by changing the lines 250-252 where the file name is concatenated.
- Defaults to "u", such that it matches a sequence of files named u.1.gmv, u.2.gmv, u.3.gmv, . . . , u.10.gmv, . . .

file name of MPEG movie (output): **-o filename**

- Basename of MPEG output (i.e. without the extension .mpeg)
- Defaults to "movie", such that a movie called movie.mpeg is created.

indices of input files: **-fls number1,number2,number3**

- The sequence of input files starts with number <number1> and ends with <number2> with a stride of <number3>. If <number3> is omitted or zero, time stepping is adaptive and all files available are taken.
  ("fls" stands for first,last,stride)
- Defaults to "1,100000,0", such every existing `GMV` file in the range 1 till 100.000 is used for creating a movie.

invisible mode: **-I, --invisible**

- Both the OpenGL and the Mesa version of `GMV` will pop up a window for each file processed and make a snapshot of it. Invoking `gmvmpeg` with this flag causes the use of the batch version of `GMV` such that the generation process is done in background mode. You will notice nothing but an increasing load of your computer and progress being made as subsequent files are processed. No X server display is needed.

  Using this option, the complete MPEG generation of one or several videos can be transferred to an arbitrary computer in a network. This means, you can even log in via modem and start this "visualization process" in a VT100 emulation.
- Not set by default.

These command line options are the ones that you will most likely change each time. The following you will probably change rarely:

keep YUV frames and RGB files: **-k, --keep-files**

- The snapshots are converted to YUV3 format and are, by default, deleted when the MPEG encoding has finished. If you want to play with different bitrates specify this option to avoid unnecessary regeneration. With the program `mkmpeg` mentioned in [Ack98] you can then play around with different movie file sizes.[2]
- Not set by default.

maximum size of MPEG movie: **-m number, --max number**

- Tells the MPEG encoder to limit the file size to <number> MB.
- Not set by default. Thus, by default there is no file size limitation.

verbose: **-V, --verbose**

- Verbose mode.
- Not set by default. Thus, by default `gmvmpeg` will swallow all output from `GMV` and the MPEG encoder.

version information: **--version**

- Prints version information.
- Not set by default.

window size: **-x number, -y number**

- Resolution in x- and y-direction of the movie to be generated.
- Defaults to a window of size 800x600.

[2]The program `mkyuv` which is also introduced in [Ack98] is obsolete upon availability of `gmvmpeg`.

Where to get the programs mentioned

(see also our homepage)

- gmvmpeg: http://www.featflow.de/download/gmvmpeg
- GMV: http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html
- NetPBM: http://wuarchive.wustl.edu/graphics/graphics/packages/NetPBM/
- ImageMagick: http://www.imagemagick.org/
- mpeg: http://www.mpeg.org/MPEG/video.html#video-software or visit the MPEG homepage at http://www.cselt.stet.it/mpeg/
- compress: shipped with every Un*x flavour
- gzip: ftp://ftp.gnu.org/pub/gnu/gzip/
- bzip2: http://sources.redhat.com/bzip2/

Example

Finally, we want to show an invocation of gmvmpeg. We will give the same example as in [Ack98] where we visualized a pressure distribution.

- Start GMV and adjust its settings for displaying a pressure distribution.
- Save your adjustments in an attribute file named “pressure.attr”.
- To create an MPEG movie called “pressure.mpeg” with a 400x320 resolution from the data files “u.1.gmv” to “u.99.gmv”, just type:

```
gmvmpeg -a pressure.attr -i u -fls 1,99 -o pressure -x 400 -y 320
```

- If you have prepared additional attribute files “streamfunction.attr” and “temperature.attr”, the batch oriented MPEG generation in “invisible” mode (i.e. with exploitation of the batch version of GMV) can be started with the following shell script (only every second file is processed):

```
#!/bin/sh
gmvmpeg -i u -fls 1,99,2 -x 400 -y 320 --invisible \
        -a pressure.attr       -o pressure

gmvmpeg -i u -fls 1,99,2 -x 400 -y 320 --invisible \
        -a streamfunction.attr -o streamfunction

gmvmpeg -i u -fls 1,99,2 -x 400 -y 320 --invisible \
        -a temperature.attr    -o temperature
# End sample.sh
```

Remarks

Unlike in the first version of gmvmpeg, there is no more need for the virtual framebuffer X server called Xvfb from the XFree86 Project, Inc.
Starting from GMV version 2.7, there is a batch version of GMV that does the rendering in background and saving it to disk. Thus, the annoying peculiarity about the visualization process in former versions of gmvmpeg that for each file GMV pops up a window on your screen and takes a screenshot of it is no longer

existing. Hence, our work-around with `Xvfb` (side-track the output of `GMV` to this X server which emulates a dumb framebuffer using virtual memory) was obsolete.

Program listing

```
#!/usr/local/bin/perl -w
#
# by Sven H.M. Buijssen, 2001/09/01

#
# gmvmpeg - script to automate generation of mpeg movies with GMV.
#
use strict;
use POSIX;
use File::Basename;
use IO::Handle;

#
# Location of programs needed
#
my $gmvGL        ="/usr/local/bin/gmvgl";          # OpenGL version of gmv
my $gmvMesa      ="/usr/local/bin/gmv";            # Mesa version of gmv
my $gmvBatch     ="/usr/local/bin/gmvbatch";       # Mesa Batch version of gmv
my $gmvToUse     =$gmvMesa;

my $sgitopnm     ="/usr/local/bin/sgitopnm";       # Convert gmv rgb screenshots to
my $ppmtoyuvsplit="/usr/local/bin/ppmtoyuvsplit";  # a format that the mpeg encoder needs.
my $mpegEncoder  ="/usr/local/bin/mpeg";           #

# Some utilities
my $uncompress   ="/usr/bin/uncompress";
my $gunzip       ="/usr/bin/gunzip";
my $bunzip2      ="/usr/local/bin/bunzip2";


#
# Some default values
#
(my $progname=$0) =~ s/^.*\/(.*)/$1/;
my $version="2.0.0";
chomp(my $host=‘hostname‘);


sub usage {
    print "Usage:
$progname [-a attribute_file] [-fls first,last,stride] [-i filename_prefix]
    [-m size] [-o output_filename] [--verbose] [-x xres] [-y yres] [--invisible]

(*) -a   : path for gmv attribute file to use
    -fls : two or three comma separated digits that specify first and
           last index of gmv input file as well as the stride
           If stride is omitted or set to 0, time step is adaptive (default).
    -h, --help:
           this help screen
    -i   : prefix of gmv input files (followed by a dot and a number without
           zero fills.)
           postfix \".gmv\" is assumed.
           e.g. u for u.2.gmv, u.3.gmv, u.4.gmv, ...
           (default: u)
    -I, --invisible:
           use gmvBatch for rendering process (invisible mode)
    -k, --keep-files:
           don’t delete temporary YUV frames
    -m, --max:
           maximum size of mpeg movie (in MB) (default: no limitation)
```

```
    -o   : name of output file (without extension .mpeg)
           (default: movie)
    -V, --verbose:
           verbose mode
    --version:
           print version information
    -wd:   set working directory for attribute, input and output files
    -x   : resolution in x direction (default: 800)
    -y   : resolution in y direction (default: 600)
    -Z   : decompress input files on-the-fly using compress
    -gz  : decompress input files on-the-fly using gzip
    -bz2 : decompress input files on-the-fly using bzip2

At least an attribute file has to be specified.

Example:
   $progname -a gmv_example.attr -o example -i u -fls 1,99,2

\n";
}


sub parseargv {
    chomp(my $pwd=‘pwd‘);

    my ($wd) = ($pwd);
    my ($attrfile, $inbasename, $outfile) = ("default.attr", "u", "movie");
    my ($first, $last, $stride, $adaptive) = (1, 100000, 1, 1);
    my ($invisibleMode, $use_uncompress, $use_gunzip, $use_bunzip2) = (0, 0, 0, 0);
    my ($keepfiles, $maxmb, $max_bits, $default_bitrate) = (0, 0, 0, 5000000);
    my ($procs) = (1);
    my ($xres, $yres) = (800, 600);
    my $verbose = "> /dev/null";
    my $ok = 0;
    my @list;

    for (my $i=0; $i<=$#ARGV; $i++) {
        my $arg=$ARGV[$i];

        if ($arg eq "-a") {
            $attrfile = $ARGV[$i+1];
            $ok++;
        } elsif ($arg eq "-fls") {
            @list = split(/,/, $ARGV[$i+1]);
            if ($#list < 1  ||  $#list > 3) {
                die "$progname: Syntax error in argument -fls, specifying\n".
                    "first and last index as well as stride.\n\n".
                    "Syntax has to be:\n   first,last,stride\n\n";
            }

            $first=$list[0];
            $last=$list[1];
            if ($#list==2) {
                $stride=$list[2];
            } else {
                $stride=0;
            }

            # adaptive time stepping if stride=0
            if ($stride==0) {
                $adaptive=1;
                $stride=1;
            }
        } elsif ($arg eq "-help" || $arg eq "--help") {
            usage();
            exit;
        } elsif ($arg eq "-i") {
```

```
            $inbasename=$ARGV[$i+1];
        } elsif ($arg eq "-I" || $arg eq "--invisible") {
            $invisibleMode=1;
            $gmvToUse=$gmvBatch;
        } elsif ($arg eq "-k" || $arg eq "--keep-files") {
            $keepfiles=1;
        } elsif ($arg eq "-m" || $arg eq "--max") {
            $maxmb=$ARGV[$i+1];
        } elsif ($arg eq "-o") {
            $outfile=$ARGV[$i+1];
        } elsif ($arg eq "-V" || $arg eq "--verbose") {
            $verbose="";
        } elsif ($arg eq "--version") {
            die "$progname version $version\n" .
                "use '$progname -h' for a list of options\n";
        } elsif ($arg eq "-wd") {
            $wd=$ARGV[$i+1];
        } elsif ($arg eq "-j") {
            $procs=int($ARGV[$i+1]);
            if ($procs < 1  ||  $procs > 8) {
                die "$progname: Error in argument -j\n".
                    "Please specify a reasonable value for the number of gmv processes\n\n";
            }
        } elsif ($arg eq "-x") {
            $xres=$ARGV[$i+1];
        } elsif ($arg eq "-y") {
            $yres=$ARGV[$i+1];
        } elsif ($arg eq "-Z") {
            $use_uncompress=1;
        } elsif ($arg eq "-gz") {
            $use_gunzip=1;
        } elsif ($arg eq "-bz2") {
            $use_bunzip2=1;
        }
    }

    if ($ok < 1) {
        usage();
        exit;
    }

    $attrfile   = "$wd/$attrfile"   unless (dirname($attrfile)   =~ m|^/|);
    $inbasename = "$wd/$inbasename" unless (dirname($inbasename) =~ m|^/|);
    $outfile    = "$wd/$outfile"    unless (dirname($outfile)    =~ m|^/|);

    return ($xres, $yres, $attrfile, $inbasename, $outfile,
            $first, $last, $stride, $adaptive,
            $invisibleMode, $use_uncompress, $use_gunzip, $use_bunzip2,
            $keepfiles, $maxmb, $max_bits, $default_bitrate, $verbose);
}


sub check_file_existence {
    my ($gmvToUse, $invisibleMode,
        $sgitopnm, $ppmtoyuvsplit, $mpegEncoder, $attrfile,
        $use_uncompress, $uncompress, $use_gunzip, $gunzip, $use_bunzip2, $bunzip2) = (@_);

    # Check for gmv program
    die "$progname: GMV program not found or not executable.\n".
        "$gmvToUse: No such file or directory/Not executable.\n".
        "Please check the values in line 16-18 of $progname\n".
        "Nothing done.\n" if (! -X $gmvToUse);

    # Check for convert programs
    die "$progname: One of the convert programs could not be found.\n".
        "$sgitopnm: No such file or directory/Not executable.\n".
        "Please check the value in line 21 of $progname\n".
```

```
        "Nothing done.\n" if (! -X $sgitopnm);
    die "$progname: One of the convert programs could not be found.\n".
        "$ppmtoyuvsplit: No such file or directory/Not executable.\n".
        "Please check the value in line 22 of $progname\n".
        "Nothing done.\n" if (! -X $ppmtoyuvsplit);

    # Check for mpeg encoder
    die "$progname: MPEG encoding program not found or not executable.\n".
        "$mpegEncoder: No such file or directory/Not executable\n".
        "Please check the value in line 23 of $progname\n".
        "Nothing done.\n" if (! -X $mpegEncoder);

    # Check whether attrib file exists and is readable
    die "$progname: Attribute file \"$attrfile\" does not exist or ".
        "is not readable.\nNothing done.\n" if (! -r $attrfile);

    # Check for utility programs when we are supposed to use them
    die "$progname: uncompress program not found or not executable.\n".
        "$uncompress: No such file or directory/Not executable\n".
        "Please check the value in line 26 of $progname\n".
        "Nothing done.\n" if ($use_uncompress  &&  ! -X $uncompress);
    die "$progname: gunzip program not found or not executable.\n".
        "$gunzip: No such file or directory/Not executable\n".
        "Please check the value in line 27 of $progname\n".
        "Nothing done.\n" if ($use_gunzip  &&  ! -X $gunzip);
    die "$progname: uncompress program not found or not executable.\n".
        "$bunzip2: No such file or directory/Not executable\n".
        "Please check the value in line 28 of $progname\n".
        "Nothing done.\n" if ($use_bunzip2  &&  ! -X $bunzip2);

    return;
}


# Catch ^C
$SIG{INT} = \&cleanExit;
STDOUT->autoflush(1);

my ($xres, $yres, $attrfile, $inbasename, $outfile,
    $first, $last, $stride, $adaptive,
    $invisibleMode, $use_uncompress, $use_gunzip, $use_bunzip2, $keepfiles,
    $maxmb, $max_bits, $default_bitrate, $verbose) = &parseargv();
print "$progname version $version\n";
&check_file_existence($gmvToUse, $invisibleMode,
                      $sgitopnm, $ppmtoyuvsplit, $mpegEncoder, $attrfile,
                      $use_uncompress, $uncompress,
                      $use_gunzip, $gunzip,
                      $use_bunzip2, $bunzip2);

my $j=0;
print "\n";
for (my $i = $first;  $i <= $last;  $i += $stride) {
    my $go = 0;
    my $filename = $inbasename . ".";
    $filename .= $i . ".gmv";
    my $displayname = $filename;

#    my $filename = $inbasename . ".t";
##   my $filename = $inbasename;
#    if ($i < 10)   {  $filename .= "0"; }
#    if ($i < 100)  {  $filename .= "0"; }
#    $filename .= $i . ".gmv";
#    my $displayname = $filename;

    # Process when file exists
    print "Testing existence of $filename(|.Z|.gz|.bz2).\n" if ($verbose eq "");
    if (-r $filename) {
```

```
        $go = 1;
    } elsif ($use_uncompress  &&  -r "$filename.Z") {
        my $tmpname = POSIX::tmpnam();
        print "$uncompress -c $filename.Z > $tmpname" if ($verbose eq "");
        system("$uncompress -c $filename.Z > $tmpname");
        $filename = $tmpname;
        $displayname .= ".Z";
        $go = 2;
    } elsif ($use_gunzip  &&  -r "$filename.gz") {
        my $tmpname = POSIX::tmpnam();
        print "$gunzip -c $filename.gz > $tmpname" if ($verbose eq "");
        system("$gunzip -c $filename.gz > $tmpname");
        $filename = $tmpname;
        $displayname .= ".gz";
        $go = 3;
    } elsif ($use_bunzip2  &&  -r "$filename.bz2") {
        my $tmpname = POSIX::tmpnam();
        print "$bunzip2 -c $filename.bz2 > $tmpname" if ($verbose eq "");
        system("$bunzip2 -c $filename.bz2 > $tmpname");
        $filename = $tmpname;
        $displayname .= ".bz2";
        $go = 4;
    }

    if ($go) {
        $j++;

        print "*** Processing $displayname ... ";
        print "\n" if ($verbose eq "");
        print "$gmvToUse -m -a $attrfile -w 0 0 $xres $yres -i $filename ".
              "-s $outfile$j.rgb $verbose\n" if ($verbose eq "");
        system("$gmvToUse -m -a $attrfile -w 0 0 $xres $yres \\
                -i $filename -s $outfile$j.rgb $verbose");

        # Convert gmv output to mpeg input format
        if ($verbose eq "") {
            print  "$sgitopnm $outfile$j.rgb | $ppmtoyuvsplit $outfile$j $verbose\n";
            system("$sgitopnm $outfile$j.rgb | \\
                    $ppmtoyuvsplit $outfile$j $verbose");
            print "File triple <$outfile$j.[YUV]> has been created.\n";
            print "*** ";
        } else {
            system("$sgitopnm $outfile$j.rgb 2>/dev/null | \\
                    $ppmtoyuvsplit $outfile$j $verbose");
        }
        print "done.\n";

        # Delete on-the-fly decompressed files.
        if ($go >= 2) {
            print "Removing $filename\n" if ($verbose eq "");
            unlink "$filename";
        }

    # If file does not exist and we don't have an adaptive time stepping
    # print error message and exit.
    } elsif (! $adaptive) {
        print "$progname: $filename: No such file or directory\n";
        exit;
    }

    print "\n" if ($verbose eq "");
}


# If at least one input file has been processed, generate a mpeg movie
if ($j > 0) {
    # Compute bitrate for mpeg encoder
```

```
    if ($maxmb <= 0) {
        $max_bits = int($default_bitrate * $j / 50);
    } else {
        $max_bits = int($maxmb * 1024 * 1024 * 8);
    }

    # Generate mpeg
    print "*** Generating movie $outfile.mpeg ... ";
    print "\n$mpegEncoder -PF -p 3 -a 1 -b $j -h $xres -v $yres ".
          "-x $max_bits -s $outfile.mpeg $outfile $verbose\n" if ($verbose eq "");
    system("$mpegEncoder -PF -p 3 -a 1 -b $j -h $xres -v $yres \\
            -x $max_bits -s $outfile.mpeg $outfile $verbose");
    print "*** " if ($verbose eq "");
    print "done.\n";

    # Clean up working directory
    if ($keepfiles == 0) {
        print "*** Removing temporary files ... ";
        for (my $jj = 0;  $jj <= $j;  $jj++) {
            unlink "$outfile$jj.rgb", "$outfile$jj.Y", "$outfile$jj.U", "$outfile$jj.V";
        }
        print "done.\n";
    }
} else {
    print "$progname: No input files found. Nothing done.\n";
}


sub cleanExit
{
  my $message;

  $message = shift || "Cancelled";
  print STDERR "$progname: $message.\n";

  exit 1;
}

exit;
# End of gmvmpeg
```

References

[Ack98] Jens F. Acker. *Working with GMV under FEATFLOW*. Preprint 98 - 50 (SFB 359), October 1998.

[AT99] Jens F. Acker and Stefan Turek. *3D Presentation of FEATFLOW Data with GMV*. Preprint 99 - 19 (SFB 359), April 1999.

[Ort01] Frank A. Ortega. *GMV 2.7. General Mesh Viewer Uses's Manual*. Los Alamos National Laboratory, 2001. www-xdiv.lanl.gov/XCM/gmv/.

[Tur96] Stefan Turek. *Finite element software for the incompressible Navier-Stokes equations: User Manual, Release 1.1*, May 1996. www.featflow.de.

All these papers are available at `http://www.featflow.de/documentation.html`.