Batch-oriented MPEG generation with GMV in background mode

Sven H.M. Buijssen, Stefan Turek

Institute of Applied Mathematics, University of Heidelberg
Im Neuenheimer Feld 294, 69120 Heidelberg, Germany
Email: featflow@gaia.iwr.uni-heidelberg.de
Homepage: www.iwr.uni-heidelberg.de/~featflow

June, 1999

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 documentation.
Mainly two features of `GMV` are exploited by the script `gmvmpeg`: `GMV` comes with the possibility to save an arbitrary configuration into an attribute file. Thus, all you need to do to generate an MPEG movie out of a sequence of files containing data from a numerical simulation is the following: (1) load a (more or less) arbitrary file from your sequence, (2) make a decision on the subset of data to be displayed and (3) save this configuration into a so-called attribute file. See also the other `GMV` reports at our homepage which describe the use of `GMV` for FEATFLOW data.
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.
But there is one annoying peculiarity about the visualization process: for each file it processes `GMV` pops up a window on your screen and takes a screenshot of it. There is (at least for us) no way to tell `GMV` to do the visualization in background. Therefore, we avail ourselves in this program of a virtual framebuffer X server called `Xvfb`. The output of `GMV` is side-tracked to this X server which emulates a dumb framebuffer using virtual memory. Hence, we have no more windows popping up disturbing our work. In fact, 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.

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).
- `sgitopnm` and `ppmtoyuvsplit` from the `netpbm` package for converting the `GMV` images into a format the MPEG encoder can handle
- `Xvfb` [1] (if you want to generate MPEG movies in background) which comes with the Xfree86 package which is distributed for free (see below).
- `mpeg` for the encoding process (see below).

Command line options

`gmvmpeg` has the following command line options to control the noninteractive generation of MPEG movies. They are partially explained in the `GMV` reports from our homepage, see also the references:

attribute file: **-a filename**
Path of the `GMV` attribute file to be used.

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.

file name of MPEG movie (output): **-o filename**
Basename of MPEG output (i.e. without the extension .mpeg)

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)

These command line options are imperative. Optional are the following:

invisible mode: **-I, --invisible**
By default `GMV` will pop up a window for each file processed and make a snapshot of it. Due to the use of `Xvfb` you will not notice the generation process. `gmvmpeg` finds out the lowest available display number on the host concerning and starts `Xvfb` at that display.

keep YUV frames: **-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 [2] you can then play around with different movie file sizes.[2]

maximum size of MPEG movie: **-m number, --max number**
Tells the MPEG encoder to limit the file size to <number> MB. By default there is no limitation.

[1] We successfully compiled `Xvfb` under SGI Irix 6.5 as well as under SunOS 5.5 and 5.6. However, we observed a latent instability: `Xvfb` crashes when `GMV` has processed approximately a dozen input files. Therefore we have chosen to relaunch it for every input file.

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

verbose: **-V, --verbose**
Verbose mode. By default `gmvmpeg` will swallow all output from `GMV` and the MPEG encoder.

version information: **--version**
Prints version information.

window size: **-x number, -y number**
Resolution in x- and y-direction of the movie to be generated. Default values are 800x600.

Where to get the programs mentioned

(see also our homepage)

- `gmvmpeg: http://www.iwr.uni-heidelberg.de/~featflow/dl.html`
- `GMV: http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html`
- `NetPBM: http://wuarchive.wustl.edu/graphics/graphics/packages/NetPBM/`
- `ImageMagick: http://www.wizards.dupont.com/cristy/ImageMagick.html`
- `Xvfb: http://www.xfree.org/`
- `mpeg: http://www.mpeg.org/MPEG/video.html#video-software` or visit the MPEG homepage at `http://www.cselt.stet.it/mpeg/`

For convenience reasons `gmvmpeg` autonomously determines the next free socket on a computer. This functionality was taken from the perl script `vncserver`, a wrapper script to start an X VNC server. This package (which is not needed for `gmvmpeg`) is published under the terms of the GNU General Public License and can be obtained from

- `Virtual Network Computing (VNC): http://www.uk.research.att.com/vnc/index.html`

Example

Finally, we want to show an invocation of `gmvmpeg`. We will give the same example as in [2] 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 `Xvfb`) 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
```

Program listing

```
#!/usr/local/bin/perl
#
# by Sven H.M. Buijssen, 99/06/14

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

#
# Location of programs needed
#
local $gmvGL        ="/usr/local/bin/gmv";         # OpenGL version of gmv
local $gmvMesa      ="/usr/local/bin/gmv.mesa";    # Mesa version of gmv (used in
                                                   # invisible mode (= use of Xvfb)
local $Xvfb         ="/usr/local/bin/Xvfb";
local $sgitopnm     ="/usr/freeware/bin/sgitopnm";
local $ppmtoyuvsplit="/usr/freeware/bin/ppmtoyuvsplit";
local $mpegEncoder  ="/usr/local/bin/mpeg";

#
# Parameter for used programs
#
local $gmvToUse     =$gmvGL;
# Typical Xvfb parameter on SGI (font path from /var/X11/xfs/config)
# (Make sure that every path listed in the font path does really exist.
#  Otherwise Xvfb won't start.)
local $XvfbParams   ="-ac -terminate -fp \"".
                     "/usr/lib/X11/fonts/100dpi/,/usr/lib/X11/fonts/75dpi/,".
                     "/usr/lib/X11/fonts/misc/,/usr/lib/X11/fonts/Type1/,".
                     "/usr/lib/X11/fonts/Speedo/\" ".
                     "-sp /usr/lib/X11/xserver/SecurityPolicy ".
                     "-screen 1 1028x768x32";
# On Sun's we use following parameters
#local $XvfbParams   ="-ac -terminate -fp \"".
#                     "/usr/openwin/lib/X11/fonts/misc,".
```

```
#                     "/usr/openwin/lib/X11/fonts/Speedo,".
#                     "/usr/openwin/lib/X11/fonts/Type1,".
#                     "/usr/openwin/lib/X11/fonts/100dpi,".
#                     "/usr/openwin/lib/X11/fonts/75dpi\" ".
#                     "-sp /usr/openwin/lib/X11/xserver/SecurityPolicy ".
#                     "-screen 1 1028x768x32 ".
#                     "-co /usr/openwin/lib/X11/rgb";


#
# Some default values
#
local $version="1.0.0";
local $default_bitrate=5000000;
(local $progname=$0) =~s/^.*\/(.*)/$1/;


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.
    -h, --help:
           this help screen
(*) -i   : prefix of gmv input files (followed by a dot and anumber without
           zero fills.)
           postfix \".gmv\" is assumed.
           e.g. u for u.2.gmv, u.3.gmv, u.4.gmv, ...
    -I, --invisible:
           use X server Xvfb for rendering process
    -k, --keep-files:
           don't delete temporary YUV frames
    -m, --max:
           maximum size of mpeg movie (in MB) (if omitted no limitation)
(*) -o   : name of output file (without extension .mpeg)
    -V, --verbose:
           verbose mode
    --version:
           print version information
    -x   : resolution in x direction (if omitted set to 800)
    -y   : resolution in y direction (if omitted set to 600)

Arguments marked with (*) have to be specified.

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

```
\n";
}


sub parseargv {
    local $ok=0;
    $xres=800;
    $yres=600;
    $attrfile="default.attr";
    $inbasename="u.";
    $outfile="movie";
    $first=1;
    $last=1;
    $stride=1;
    $adaptive=0;
    $keepfiles=0;
    $maxmb=0;
    $verbose="> /dev/null";
    $useXvfb=0;

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

        if ($arg eq "-a") {
            $attrfile=$ARGV[$i+1];
            $ok++;
        } elsif ($arg eq "-fls") {
            @list=split(/,/,$ARGV[$i+1]);
            if ($#list<1 || $#list>3) {
                print "$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";
                exit;
            }

            $first=$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;
            }
            $ok++;
```

```
        } elsif ($arg eq "-help" || $arg eq "--help") {
            usage();
            exit;
        } elsif ($arg eq "-i") {
            $inbasename=$ARGV[$i+1];
            $ok++;
        } elsif ($arg eq "-I" || $arg eq "--invisible") {
            $useXvfb=1;
            $gmvToUse=$gmvMesa;
        } 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];
            $ok++;
        } elsif ($arg eq "-V" || $arg eq "--verbose") {
            $verbose="";
        } elsif ($arg eq "--version") {
            print "$progname version $version\n" .
                  "use '$progname -h' for a list of options\n";
            exit;
        } elsif ($arg eq "-x") {
            $xres=$ARGV[$i+1];
        } elsif ($arg eq "-y") {
            $yres=$ARGV[$i+1];
        }
    }

    if ($ok<4) {
        usage();
        exit;
    }
}

parseargv();
STDOUT->autoflush(1);

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

if ($useXvfb == 1) {
    $originalDisplay = $ENV{DISPLAY};
```

```
    &GetSocketConstants();
    $XvfbDisplayNumber = &GetDisplayNumber();

    ($XvfbDisplay = $originalDisplay) =~s/(^.*):.*/$1:$XvfbDisplayNumber.0/;
    $ENV{DISPLAY} = $XvfbDisplay;
}

local $j=0;
for ($i=$first; $i<=$last; $i+=$stride) {
    local $filename=$inbasename . "." . $i . ".gmv";

    # Process when file exists
    if (-r $filename) {
        $j++;

        # If specified start Xvfb
        if ($useXvfb == 1) {
            system("$Xvfb :$XvfbDisplayNumber $XvfbParams $verbose 2>/dev/null &");
        }

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

        system("$gmvToUse -m -a $attrfile -w 0 0 $xres $yres \\
                -i $filename -s $verbose");
        if ($verbose eq "") {
            system("$sgitopnm AzsnapgmvAz | \\
                    $ppmtoyuvsplit $outfile$j $verbose");
            print "*** ";
        } else {
            system("$sgitopnm AzsnapgmvAz 2>/dev/null | \\
                    $ppmtoyuvsplit $outfile$j $verbose");
        }
        print "done.\n";

    # 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;
    }
}


# If at least one input file has been processed, generate a mpeg movie
if ($j > 0) {
    # Reset DISPLAY environment variable
    $ENV{DISPLAY} = $originalDisplay if ($useXvfb == 1);
```

```
    # Compute bitrate for mpeg encoder
    if ($maxmb <= 0) {
        $max_bits = $default_bitrate * $j / 25;
    } else {
        $max_bits = $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... ";
        system("rm AzsnapgmvAz $verbose");
        system("rm $outfile*.Y");
        system("rm $outfile*.U");
        system("rm $outfile*.V");
        print "done.\n";
    }
} else {
    print "$progname: No input files found. Nothing done.\n";
}


##########################################################################
#
# Stuff taken from
# vncserver - wrapper script to start an X VNC server.
#
# http://www.uk.research.att.com/vnc/index.html
##########################################################################

#
# Find socket constants. 'use Socket' is a perl5-ism, so we wrap it in an
# eval, and if it fails we try 'require "sys/socket.ph"'.  If this fails,
# we just guess at the values.  If you find perl moaning here, just
# hard-code the values of AF_INET and SOCK_STREAM.  You can find these out
# for your platform by looking in /usr/include/sys/socket.h and related
# files.
#

sub GetSocketConstants
{
    chop($os = `uname`);
```

```
    chop($osrev = `uname -r`);

    eval 'use Socket';
    if ($@) {
eval 'require "sys/socket.ph"';
if ($@) {
    if (($os eq "SunOS") && ($osrev !~ /^4/)) {
$AF_INET = 2;
$SOCK_STREAM = 2;
    } else {
$AF_INET = 2;
$SOCK_STREAM = 1;
    }
} else {
    $AF_INET = &AF_INET;
    $SOCK_STREAM = &SOCK_STREAM;
}
    } else {
$AF_INET = &AF_INET;
$SOCK_STREAM = &SOCK_STREAM;
    }
}

#
# GetDisplayNumber gets the lowest available display number.
#

sub GetDisplayNumber
{
    chop($host = `uname -n`);
    foreach $n (1..99) {
        if (&CheckDisplayNumber($n)) {
            return $n;
        }
    }

    die "$prog: no free display number on $host.\n";
}


#
# CheckDisplayNumber checks if the given display number is available.
#

sub CheckDisplayNumber
{
    local ($n) = @_;

    socket(S, $AF_INET, $SOCK_STREAM, 0) || die "$prog: socket failed: $!\n";
    if (!bind(S, pack('S n x12', $AF_INET, 6000 + $n))) {
```

```
        close(S);
        return 0;
    }
    close(S);

    socket(S, $AF_INET, $SOCK_STREAM, 0) || die "$prog: socket failed: $!\n";
    if (!bind(S, pack('S n x12', $AF_INET, 5900 + $n))) {
        close(S);
        return 0;
    }
    close(S);

    if (-e "/tmp/.X$n-lock") {
        warn "\nWarning: $host:$n is taken because of /tmp/.X$n-lock\n";
        warn "Remove this file if there is no X server $host:$n\n";
        return 0;
    }

    if (-e "/tmp/.X11-unix/X$n") {
        warn "\nWarning: $host:$n is taken because of /tmp/.X11-unix/X$n\n";
        warn "Remove this file if there is no X server $host:$n\n";
        return 0;
    }

    return 1;
}


exit;
# End of gmvmpeg
```

References

[1] Finite element software for the incompressible Navier-Stokes equations: User Manual, Release 1.1, 1998 (see also: `http://www.iwr.uni-heidelberg.de/~featflow/`)

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

[3] J.F. Acker, S. Turek, 3D Presentation of FEATFLOW Data with GMV, Preprint 99 - 19 (SFB 359), April 1999

All these papers are available at `http://www.iwr.uni-heidelberg.de/~featflow/dl.html`.