

# Batch-oriented MPEG generation with GMV in background mode

Release 2.5x

Sven H.M. Buijssen, Stefan Turek

Institute of Applied Mathematics, University of Dortmund  
Vogelpothsweg 87, 44227 Dortmund, Germany  
Email: [featflow@featflow.de](mailto:featflow@featflow.de)  
Homepage: [www.featflow.de](http://www.featflow.de)

May 2005

## Abstract

For automatisisation 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.<sup>1</sup>
4. Save your settings into a so-called attribute file.
5. Finally run the perl-script `gmvmpeg` with appropriate command line 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 GMVfiles 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.
- Default: "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.

---

<sup>1</sup>The manuals [Ack98], [AT99] and especially the complete GMVmanual [Ort01] will help you withal.

- Default: "u"  
(Thus, a sequence of files named u.1.gmv, u.2.gmv, u.3.gmv, ..., u.10.gmv, ... is matched.)

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

- Basename of MPEG output (i.e. without the extension .mpeg)
- Default: "movie"  
(Thus, a movie called movie.mpeg is created.)

indices of input files: **-f1s 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.  
("f1s" stands for first,last,stride)
- Default: "1,100000,0"  
(Thus, 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.
- Default: not set.

creation of MPEG-1 or MPEG-2 files: **--mpeg1, --mpeg2**

- Use MPEG-1 or MPEG-2 encoder respectively to convert the sequence of GMV snapshots into a movie.
- Default: MPEG-1

on-the-fly decompression of files: **-Z, -gz, -bz2**

- Look for GMV files with suffix .Z, .gz or .bz2 and decompress them on-the-fly using `compress`, `gzip` and `bzip2` respectively.
- Default: not set.

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

help screen: **-help**

- Show a help screen explaining all possible command line options.

check paths: **--checkonly**

- Verify that all program paths are set correctly within the script.

bitrate: **-b, --bitrate**

- Specify bitrate for the MPEG movie.
- Default: 5000000 for MPEG-1, 3500000 for MPEG-2

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.<sup>2</sup>
- Default: not set.

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

- Obsolete. Just provided for compatibility reasons. Use `-b/--bitrate` instead.
- Tells the MPEG encoder to limit the file size to `<number>` MB.
- Default: not set.  
(Thus, by default there is no file size limitation.)

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

- Verbose mode.
- Default: not set.  
(Thus, by default `gmvmpeg` will swallow all output from GMV and the MPEG encoder.)

version information: **--version**

- Prints version information.
- Default: not set.

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

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

---

<sup>2</sup>The program `mkycuv` 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/bin/env perl
#
# by Sven H.M. Buijssen

#
# gmvmpeg - script to automate generation of mpeg movies with GMV.
#
# History:
# 2005/05/17   v2.5.5   - Release version for FeatFlow CD 1.3
# 2005/01/14   v2.5.4   - Check whether horizontal and vertical
#                        resolution are even.
#
#                        - New option '--checkonly'
# 2004/05/28   v2.5.3   - When decompressing on-the-fly, show file
#                        name before decompressing
#
#                        - Do not hard-code location of perl in
#                        first line, but use perl from $PATH
# 2004/05/28   v2.5.2   - Fixed bugs concerning automatic file
#                        deletion and MPEG-2 bit rate setting
# 2004/05/28   v2.5.1   - Added support for MPEG-2 movies with
#                        bigger resolutions than 720x576
# 2004/01/28   v2.5.0   - Added support for MPEG-2 encoding
# ...
# ...           (Undocumented changes)
# ...
# 2002/././..   v1.?.?. - Added decompressing on-the-fly
# 2002/././..   v1.?.?. - Changes to script to reflect the changes
#                        in GMV (availability of standalone
#                        gmvbatach, no snapshots with gmvm itself
#                        possible any more)
# 2001/09/01   v1.0.0   - First release
#

use strict;
use POSIX;
use File::Basename;
use IO::Handle;

#
# Location of programs needed
#
# OpenGL version of gmvm
my $gmvmGL      = "/usr/local/bin/gmvmgl";
# Mesa version of gmvm
my $gmvmMesa    = "/usr/local/bin/gmvm";
# Mesa Batch version of gmvm
my $gmvmBatch   = "/usr/local/bin/gmvmbatch";
```

```

my $gmvtToUse      = $gmvmesa;

# Convert gm v rgb screenshots to a format that the mpeg encoder needs.
my $sgitopnm       = "/usr/local/netpbm/bin/sgitopnm";
my $ppmttoyuvsplit = "/usr/local/netpbm/bin/ppmttoyuvsplit";
# MPEG-1 encoder (shipped e.g. with FeatFlow)
my $mpegEncoder    = "/usr/local/bin/mpeg";
# MPEG-2 encoder (google for mpeg2vidcodec_v12.tar.gz)
my $mpeg2Encoder   = "/usr/local/bin/mpeg2encode";

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

#
# Some default values
#
(my $progname=$0) =~ s/^.*/(.*)/$1/;
my $version="2.5.5";
chomp(my $host=`hostname`);

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

my ($xres, $yres, $attrfile, $inbasename, $outfile,
    $first, $last, $stride, $adaptive,
    $invisibleMode, $use_mpeg2, $use_uncompress, $use_gunzip, $use_bunzip2, $keepfiles,
    $maxmb, $max_bits, $bitrate, $verbose, $checkonly) = &parseargv();
print "$progname version $version\n";
&check_file_existence($gmvtToUse, $invisibleMode,
    $sgitopnm, $ppmttoyuvsplit, $mpegEncoder, $attrfile,
    $use_mpeg2, $mpeg2Encoder,
    $use_uncompress, $uncompress,
    $use_gunzip, $gunzip,
    $use_bunzip2, $bunzip2);
exit if ($checkonly);

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

    # FEAST syntax
    # my $filename = $inbasename . ".";
    # if ($i < 10) { $filename .= "0"; }
    # if ($i < 100) { $filename .= "0"; }
    # $filename .= $i . ".gm v";
    # my $displayname = $filename;

    # parpp3d++ syntax
    my $filename = $inbasename . ".t";
    # if ($i < 10) { $filename .= "0"; }
    # if ($i < 100) { $filename .= "0"; }
    # $filename .= $i . ".gm v";

```

```

#   my $displayname = $filename;

# Process when file exists
print "Testing existence of $filename(|.Z|.gz|.bz2)." if ($verbose eq "");
if (-r $filename) {
    print "*** Processing $displayname ... ";
    print "\n" if ($verbose eq "");
    $go = 1;
} elsif ($use_uncompress && -r "$filename.Z") {
    $displayname .= ".Z";
    print "*** Processing $displayname ... ";
    print "\n" if ($verbose eq "");

    my $tmpname = POSIX::tmpnam();
    print "\n$uncompress -c $filename.Z > $tmpname" if ($verbose eq "");
    system("$uncompress -c $filename.Z > $tmpname");
    $filename = $tmpname;
    $go = 2;
} elsif ($use_gunzip && -r "$filename.gz") {
    $displayname .= ".gz";
    print "*** Processing $displayname ... ";
    print "\n" if ($verbose eq "");

    my $tmpname = POSIX::tmpnam();
    print "\n$gunzip -c $filename.gz > $tmpname" if ($verbose eq "");
    system("$gunzip -c $filename.gz > $tmpname");
    $filename = $tmpname;
    $go = 3;
} elsif ($use_bunzip2 && -r "$filename.bz2") {
    $displayname .= ".bz2";
    print "*** Processing $displayname ... ";
    print "\n" if ($verbose eq "");

    my $tmpname = POSIX::tmpnam();
    print "\n$bunzip2 -c $filename.bz2 > $tmpname" if ($verbose eq "");
    system("$bunzip2 -c $filename.bz2 > $tmpname");
    $filename = $tmpname;
    $go = 4;
}

if ($go) {
    $j++;
    print "$gmvtToUse -m -a $attrfile -w 0 0 $xres $yres -i $filename ".
        "-s $outfile$j.rgb $verbose\n" if ($verbose eq "");
    system("$gmvtToUse -m -a $attrfile -w 0 0 $xres $yres \\\
        -i $filename -s $outfile$j.rgb $verbose") &&
die "\n\n$progname: $gmvtToUse has terminated unexpectedly.\n" .
    "There is no image data that can be converted into a movie.\n\n" .
    "A few guesses (order according to decreasing probability):\n" .
    "** You have pressed Ctrl-C.\n" .
    "** $gmvtToUse crashed. Possibly because of an attribute file\n" .
    "   that was created with a newer version of gmvt than the one\n" .
    "   used in $progname. Check version numbers!\n";

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

```

```

        system("$sgitopnm $outfile$j.rgb 2>/dev/null | \\\
            $ppmtouyvsplit $outfile$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) {
    # For MPEG-2 generation
    my $tmpname = POSIX::tmpnam() if ($use_mpeg2);

    # Generate mpeg
    if (! $use_mpeg2) {
# MPEG-1 generation

# Compute bitrate for mpeg encoder
if ($maxmb <= 0) {
    $max_bits = int($bitrate * $j / 50);
} else {
    $max_bits = int($maxmb * 1024 * 1024 * 8);
}
# obsolete, not used in this version

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";
    } else {
# MPEG-2 generation
print "*** Creating parameter file for MPEG-2 encoder ... ";
&create_mpeg2_config_file($tmpname, $outfile . "%d", $j, $xres, $yres, $bitrate);
print "*** \n*** Parameter file <$tmpname> created.\n" if ($verbose eq "");
print "done.\n" unless ($verbose eq "");
print "*** Generating movie $outfile.m2v ... ";
if ($verbose eq "") {
    print "\n$mpeg2Encoder $tmpname $outfile.m2v\n";
    system("$mpeg2Encoder $tmpname $outfile.m2v");
    print "*** ";
} else {
    system("$mpeg2Encoder $tmpname $outfile.m2v 2>/dev/null");
}
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$jj.V";
    }
}
unlink "$tmpname", "stat.out" if ($use_mpeg2);
print "done.\n";
}

print "\nNote: You might consider creating movies in MPEG-2 format which\n" .
"are usually smaller and of better quality.\n" .
"Just use the command line flag '--mpeg2'\n" if (! $use_mpeg2);

print "\nNote: You might consider using the 'invisible mode' for the rendering\n" .
"process. There would be no more GMV windows popping up all the time!\n" .
"Just use the command line flag '-I' or '--invisible'\n" if (! $invisibleMode);
} else {
    print "$progname: No input files found. Nothing done.\n";
}

sub usage {
    print "Usage:
$progname [-a attribute_file] [-i filename_prefix] [-fls first,last,stride]
          [-o output_filename] [--invisible] [--mpeg2] [-x xres] [-y yres] [--verbose]

(*) -a    : path for gmv attribute file to use
    -b, --bitrate:
        Specifies bitrate for MPEG movie.
        (default: 5000000 for MPEG-1, 3500000 for MPEG-2)
    -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:
        Obsolete. Only available for compatibility reasons
        Has been used to specify maximum size of mpeg movie.
        Use -b/--bitrate instead now.
    --mpeg1:
        Use MPEG-1 encoder (default).
    --mpeg2:
        Use MPEG-2 encoder instead of MPEG-1.
    -o     : name of output file (without extension .mpeg)
        (default: movie)
    -V, --verbose:
        verbose mode
    --version:
        print version information
    --checkonly:
        verify that all program paths are correct and exit

```

```

-wd:  set working directory for attribute, input and output files
-x   : horizontal resolution (default: 800 for MPEG-1, 720 for MPEG-2)
-y   : vertical resolution (default: 600 for MPEG-1, 576 for MPEG-2)
-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

```

Attributes marked with (\*) are mandatory, so 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 {
# I should really convert this routine to use Getopt::Long!!
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_mpeg2, $use_uncompress, $use_gunzip, $use_bunzip2) = (0, 0, 0, 0, 0);
my ($keepfiles, $maxmb, $max_bits, $bitrate) = (0, 0, 0, 0);
my ($procs) = (1);
my ($xres, $yres) = (0, 0);
my $verbose = "> /dev/null";
my $ok = 0;
my $checkonly = 0;
my @list;

ARGS: 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;
        }
    }

    $i++;
}

```

```

    } elif ($arg eq "-help" || $arg eq "--help") {
        usage();
        exit;
    } elif ($arg eq "-i") {
        $inbasename=$ARGV[$i+1];
    } elif ($arg eq "-I" || $arg eq "--invisible") {
        $invisibleMode=1;
        $gmvtToUse=$gmvtBatch;
    } elif ($arg eq "-k" || $arg eq "--keep-files") {
        $keepfiles=1;
    } elif ($arg eq "-m" || $arg eq "--max") {
#       $maxmb=$ARGV[$i+1];
        $i++;
    } elif ($arg eq "-b" || $arg eq "--bitrate") {
        $bitrate=$ARGV[$i+1];
        $i++;
    } elif ($arg eq "--mpeg2") {
        $use_mpeg2=1;
    } elif ($arg eq "--mpeg1") {
        $use_mpeg2=0;
    } elif ($arg eq "-o") {
        $outfile=$ARGV[$i+1];
        $i++;
    } elif ($arg eq "-V" || $arg eq "--verbose") {
        $verbose="";
    } elif ($arg eq "--version") {
        die "$progname version $version\n" .
            "use '$progname -h' for a list of options\n";
    } elif ($arg eq "-wd") {
        $wd=$ARGV[$i+1];
    } elif ($arg eq "-j") {
# parallel visualisation mode is experimental, not working yet
        $procs=int($ARGV[$i+1]);
        if ($procs < 1 || $procs > 8) {
            die "$progname: Error in argument -j\n".
                "Please specify a reasonable value for the number of gmvt processes\n\n";
        }
        $i++;
    } elif ($arg eq "-x") {
        $xres=$ARGV[$i+1];
        $i++;
    } elif ($arg eq "-y") {
        $yres=$ARGV[$i+1];
        $i++;
    } elif ($arg eq "-Z") {
        $use_uncompress=1;
    } elif ($arg eq "-gz") {
        $use_gunzip=1;
    } elif ($arg eq "-bz2") {
        $use_bunzip2=1;
    } elif ($arg eq "--checkonly") {
        $checkonly = 1;
    }
    $ok = 1;
    last ARGS;
}

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

```

```

# Check if the x- and y-resolution are conforming to MPEG-1 and -2 standard.
# Otherwise
# * the MPEG-1 encoder will produce a 19bit sized movie and
# * the MPEG-2 encoder will die with an error message
# So, in any case, all the gmV visualisation has been in vain.
if ($xres % 2) {
die "$progname: Horizontal size must be an even (4:2:0 / 4:2:2) for MPEG-1 and -2 movies.\n";
} elsif ($yres % 2) {
die "$progname: Vertical size must be an even (4:2:0 / 4:2:2) for MPEG-1 and -2 movies.\n";
}

$xres = 800 if ($xres == 0 && ! $use_mpeg2);
$yres = 600 if ($yres == 0 && ! $use_mpeg2);
$bitrate = 5000000 if ($bitrate == 0 && ! $use_mpeg2);
$xres = 720 if ($xres == 0 && $use_mpeg2);
$yres = 576 if ($yres == 0 && $use_mpeg2);
$bitrate = 3500000 if ($bitrate == 0 && $use_mpeg2);
$attrfile = "$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_mpeg2, $use_uncompress, $use_gunzip, $use_bunzip2,
        $keepfiles, $maxmb, $max_bits, $bitrate, $verbose, $checkonly);
}

sub check_file_existence {
my ($gmVToUse, $invisibleMode,
    $sgitopnm, $ppmtoyuvsplit, $mpegEncoder, $attrfile,
    $use_mpeg2, $mpeg2Encoder, $use_uncompress, $uncompress, $use_gunzip, $gunzip, $use_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 $progname\n".
    "Nothing done.\n" if (!$use_mpeg2 && ! -X $mpegEncoder);

# Check for mpeg2 encoder
die "$progname: MPEG-2 encoding program not found or not executable.\n".
    "$mpeg2Encoder: No such file or directory/Not executable\n".
    "Please check the value in line 24 of $progname\n".
    "Nothing done.\n" if ($use_mpeg2 && ! -X $mpeg2Encoder);

# 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 27 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 28 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 29 of $progname\n".
    "Nothing done.\n" if ($use_bunzip2 && ! -X $bunzip2);

return;
}

sub create_mpeg2_config_file()
{
    my ($filename, $name_source_files, $number_frames, $xres, $yres, $bitrate) = (@_);

    open (PARAMFILE, ">", $filename) or
die "\n\n$progname: Can not open <$filename>: $!\n" .
    "This file is needed for invocation of MPEG-2 encoder.\n";

    print PARAMFILE
qq{parameter file for MPEG-2 movie (PAL) from GMV files - created by gmvmpeg $version\n} .
qq{$name_source_files /* name of source files */\n} .
qq{- /* name of reconstructed images ("-": do not store) */\n} .
qq{- /* name of intra quant matrix file ("-": default matrix) */\n} .
qq{- /* name of non intra quant matrix file ("-": default matrix) */\n} .
qq{stat.out /* name of statistics file ("-": stdout) */\n} .
qq{0 /* input picture file format: 0=Y,U,V, 1=yuv, 2=ppm */\n} .
qq{$number_frames /* number of frames */\n} .
qq{1 /* number of first frame */\n} .
qq{00:00:00:00 /* timecode of first frame */\n} .
qq{12 /* N (# of frames in GOP) */\n} .
qq{3 /* M (I/P frame distance) */\n} .
qq{0 /* ISO/IEC 11172-2 stream */\n} .
qq{0 /* 0:frame pictures, 1:field pictures */\n} .
qq{$xres /* horizontal_size */\n} .
qq{$yres /* vertical_size */\n} .
qq{2 /* aspect_ratio_information 1=square pel, 2=4:3, 3=16:9, 4=2.11:1 */\n} .
qq{3 /* frame_rate_code 1=23.976, 2=24, 3=25, 4=29.97, 5=30 frames/sec. */\n} .
qq{$bitrate /* bit_rate (bits/s) */\n} .
qq{112 /* vbv_buffer_size (in multiples of 16 kbit) */\n} .
qq{0 /* low_delay */\n} .
qq{0 /* constrained_parameters_flag */\n} .
qq{4 /* Profile ID: Simple = 5, Main = 4, SNR = 3, Spatial = 2, High = 1 */\n} .
get_level_id($xres, $yres) .
qq{ /* Level ID: Low = 10, Main = 8, High 1440 = 6, High = 4 */\n} .
qq{0 /* progressive_sequence */\n} .
qq{1 /* chroma_format: 1=4:2:0, 2=4:2:2, 3=4:4:4 */\n} .
qq{1 /* video_format: 0=comp., 1=PAL, 2=NTSC, 3=SECAM, 4=MAC, 5=unspec. */\n} .
qq{5 /* color_primaries */\n} .
qq{5 /* transfer_characteristics */\n} .
qq{5 /* matrix_coefficients */\n} .
qq{$xres /* display_horizontal_size */\n} .

```

```

qq{$yres          /* display_vertical_size */\n} .
qq{0             /* intra_dc_precision (0: 8 bit, 1: 9 bit, 2: 10 bit, 3: 11 bit */\n} .
qq{1             /* top_field_first */\n} .
qq{0 0 0         /* frame_pred_frame_dct (I P B) */\n} .
qq{0 0 0         /* concealment_motion_vectors (I P B) */\n} .
qq{1 1 1         /* q_scale_type (I P B) */\n} .
qq{1 0 0         /* intra_vlc_format (I P B)*/\n} .
qq{0 0 0         /* alternate_scan (I P B) */\n} .
qq{0             /* repeat_first_field */\n} .
qq{0             /* progressive_frame */\n} .
qq{0             /* P distance between complete intra slice refresh */\n} .
qq{0             /* rate control: r (reaction parameter) */\n} .
qq{0             /* rate control: avg_act (initial average activity) */\n} .
qq{0             /* rate control: Xi (initial I frame global complexity measure) */\n} .
qq{0             /* rate control: Xp (initial P frame global complexity measure) */\n} .
qq{0             /* rate control: Xb (initial B frame global complexity measure) */\n} .
qq{0             /* rate control: d0i (initial I frame virtual buffer fullness) */\n} .
qq{0             /* rate control: d0p (initial P frame virtual buffer fullness) */\n} .
qq{0             /* rate control: d0b (initial B frame virtual buffer fullness) */\n} .
qq{2 2 11 11     /* P:  forw_hor_f_code forw_vert_f_code search_width/height */\n} .
qq{1 1 3 3       /* B1: forw_hor_f_code forw_vert_f_code search_width/height */\n} .
qq{1 1 7 7       /* B1: back_hor_f_code back_vert_f_code search_width/height */\n} .
qq{1 1 7 7       /* B2: forw_hor_f_code forw_vert_f_code search_width/height */\n} .
qq{1 1 3 3       /* B2: back_hor_f_code back_vert_f_code search_width/height */\n};

    close (PARAMFILE);

    return;
}

sub get_level_id {
    # This function finds the appropriate "level id" (mpeg2 encoding parameter)
    # for a given resolution. In case the recommended resolution for
    # the highest possible level (coded as '4') is exceeded, still '4'
    # is returned.
    #
    # The resolution limits are coded in separate arrays and can be
    # adjusted easily if necessary.
    my ($X, $Y)=(@_);

    my @Level_ID = (10, 8, 6, 4);
    my @X_Bound  = (352, 720, 1440, 1920);
    my @Y_Bound  = (240, 480, 960, 1080);

    my $i=0;
    $i++ while ($X > $X_Bound[$i]) and $i < $#X_Bound;
    $i++ while ($Y > $Y_Bound[$i]) and $i < $#Y_Bound;
    return $Level_ID[$i];
}

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/](http://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](http://www.featflow.de).
- All these papers are available at <http://www.featflow.de/documentation.html>.