Berkeley MPEG-1 Video Encoder

User's Guide

Kevin L. Gong
Computer Science Division
University of California
Berkeley, California 94720

keving@cs.berkeley.edu

Contents

Last Updated: 14 March 1994

0. Contacts

Use the following e-mail addresses to reach us:
mpeg-bugs@cs.berkeley.edu (bug reports)
keving@cs.berkeley.edu (questions)
larry@cs.berkeley.edu (funding!)

1. Installation

To install, read the directions in doc/INSTALL.

Note that the `bin/` directory contains binaries for several different platforms. The program has been successfully ported to the following platforms:

SunOS 4.x
DEC Alpha 3000/400
DECstation 5000
HP 9000 series

If you are successful in porting to a new platform, or have problems installing, please let us know.

2. Sequential Usage

The encoder is invoked in the following manner:

mpeg_encode <options> parameter_file

Here is a description of the different command-line options available and parameter-file options available in sequential (one-machine) encoding. You should definitely read sections 2.1-2.8. The other sections are optional.

In the following, whenever there is a space in the parameter file, this can be any amount of whitespace (tabs or spaces).

2.1 QScale (parameter file)

The quantization scale values (qscale) give a trade-off between quality and compression. Using different Qscale values has very little effect on speed. The qscale values can be set separately for I, P, and B-frames.

Usage:

```
IQSCALE num
PQSCALE num
BQSCALE num
```

num in all three cases is a number from 1 to 31.

Larger numbers give better compression, but worse quality.

Flower garden tests:

Table 1: Q-Scale vs. Quality

Q-Scale	I-Frames	P-Frames	B-Frames
1	43.2	46.3	46.5
6	32.6	34.6	34.3
11	28.6	29.5	30.0
16	26.3	26.8	28.6
21	24.7	25.0	27.9
26	23.5	23.9	27.5
31	22.6	23.0	27.3

Table 2: Q-Scale vs. Compression

Q-Scale	I-Frames	P-Frames	B-Frames
1	2:1	2:1	2:1
6	7	10	15
11	11	18	43
16	15	29	97
21	19	41	173
26	24	56	256
31	28	73	330

Should show graph of compression vs. quality.

2.2 Search Techniques (parameter file)

There are several different motion vector search techniques available. There are different techniques available for P-frame search and B-frame search. Using different search techniques present little difference in quality, but a large difference in compression and speed.

There are 4 types of P-frame search: Exhaustive, TwoLevel, SubSample, and Logarithmic.

There are 3 types of B-frame search: Exhaustive, Cross2, and Simple.

The suggested search techniques are TwoLevel and Logarithmic for P-frame search, and Cross2 and Simple for B-frame search. Here are some numbers comparing the different search methods:

Table 3: P-frame Motion Vector Search (Normalized)

Technique	Compression[a]	Speed[b]	Quality[c]
Exhaustive	1000	1000	1000
SubSample	1008	2456	1000
TwoLevel	1009	3237	1000
Logarithmic	1085	8229	998

a. Smaller numbers are better compression
b. Larger numbers mean faster execution
c. Larger numbers mean better quality

Table 4: B-frame Motion Vector Search (Normalized)

Technique	Compression	Speed	Quality
Exhaustive	1000	0	1000
Cross2	975	1000	996
Simple	938	1765	991

For some reason Simple seems to give better compression, but it depends on the image sequence.

Usage:

```
PSEARCH_ALG ptechnique
BSEARCH_ALG btechnique
```

ptechnique is one of {LOGARITHMIC, SUBSAMPLE, TWOLEVEL, EXHAUSTIVE}

btechnique is one of {EXHAUSTIVE, CROSS2, SIMPLE}

2.3 GOP (parameter file)

A Group of Pictures (GOP) is a roughly independently decodable sequence of frames. An MPEG video stream is made of one or more GOP's. You may specify how many frames each GOP should be. A GOP must contain at least one I-frame, and the encoder will enforce that by taking your number as the *minimum* number of frames in a GOP.

Usage:

```
GOP_SIZE num
```

where `num` = the number of frames in a GOP

2.4 Slice (parameter file)

A slice is an independently decodable unit in a frame. It can be as small as one macroblock, or it can be as big as the entire frame. Barring transmission error, adding slices does not change quality or speed; the only effect is slightly worse compression. More slices are used for noisy transmission so that errors are more recoverable. Since usually errors are not such a problem, we usually just use one slice per frame.

Usage:

```
SLICES_PER_FRAME num
```

where `num` = the number of slices in a frame

Note: Some MPEG playback systems require that each slice must consist of whole rows of macroblocks. If this is the case, then if the height of the image is H pixels, then you should set the SLICES_PER_FRAME to some number which divides H/16. For example, if H = 240, then you should only use SLICES_PER_FRAME values of 15, 5, 3, or 1.

Note to the note: these MPEG playback systems are really at fault, since the MPEG standard says this doesn't have to be so.

2.5 Search Window (parameter file)

The search window is the window in which motion vectors are searched for. The window is a square. You can specify the size of the square, and whether to allow half-pixel motion vectors or not.

Usage:

```
PIXEL <FULL or HALF>
RANGE num
```

`HALF` means that half-pixel vectors are allowed. The search window is +/- `num` pixels in the X and Y directions. It is usually important that you use `HALF`, because it results in both better quality and better compression.

`num` should probably be set to at least 8 or 10. This number depends on the image. Using much larger numbers such as 20 or 30 doesn't seem to help much, and increases the CPU cost drastically.

2.6 IPB Pattern

You can set the sequence of I, P, and B-frames. Later versions will allow you to do more than set a repeating IPB pattern. The pattern affects speed, quality, and compression. Here is a chart which shows some of the trade-offs:

Table 5: Comparison of I/P/B-Frames (Normalized)

Frame Type	Compression	Speed	Quality
I-frames	1000	1000	1000
P-frames	409	601	969
B-frames	72	260	919

(this is given a certain Q-scale)

A standard sequence is IBBPBBPBBPBBPBB.

Usage (parameter file):

```
PATTERN <IPB pattern>
```

Note that if the last frame is a B-frame, it will not be encoded (since it has no future frame to reference from). To insure that every frame is encoded, the encoder can force the last frame to be an I-frame.

Usage:

```
FORCE_ENCODE_LAST_FRAME
```

2.7 Specifying Input Files

The encoder can accept three base types of input files: PPM, PNM, and YUV. Note that PPM is a subset of

PNM; the PPM option is available because it is faster to read if the files are known to be PPM. The YUV format is described in a separate document (doc/INPUT.FORMAT).

If you use YUV format, you must specify the pixel size of the image in the parameter file.

Usage:
```
BASE_FILE_FORMAT format
YUV_SIZE widthxheight
```

`format` is one of {`YUV, PPM, PNM, JPEG`}
`width` and `height` are integers

You must specify the directory in which the input files are located. You can use '.' to specify the current directory.

Usage:
```
INPUT_DIR directory
```

You must also specify the names of the files themselves. You list them sequentially, one per line, in display order. There are shortcuts, however, which allow you to condense many files into one line.

Usage:

INPUT
$\texttt{file}_1$
$\texttt{file}_2$
...
$\texttt{file}_n$
END_INPUT

$\texttt{file}_i$ can be either a file name or a single-star expression followed by a bracketed expansion for the star. There are two types of bracketed expansions. For example:

```
sflowg.*.yuv [0-10]
```

is expanded to:

```
sflowg.0.yuv
sflowg.1.yuv
sflowg.2.yuv
sflowg.3.yuv
sflowg.4.yuv
sflowg.5.yuv
sflowg.6.yuv
sflowg.7.yuv
sflowg.8.yuv
sflowg.9.yuv
sflowg.10.yuv
```

```
sflowg.*.yuv [0-10+3]
```

is expanded to:

```
sflowg.0.yuv
sflowg.3.yuv
sflowg.6.yuv
sflowg.9.yuv
```

Also, the encoder will pad with 0's if necessary:

```
sflowg.*.yuv [00-10]
```

is expanded to:

```
sflowg.00.yuv
sflowg.01.yuv
sflowg.02.yuv
sflowg.03.yuv
sflowg.04.yuv
sflowg.05.yuv
sflowg.06.yuv
sflowg.07.yuv
sflowg.08.yuv
sflowg.09.yuv
sflowg.10.yuv
```

The encoder allows you to use other file formats by providing an input conversion specifier. You must describe how to convert the input format into one of the base file types.

```
Usage:

INPUT_CONVERT conversion
```

`conversion` must be a multi-star expression. If `conversion` is simply '*', then no conversion takes place. Otherwise, each of the file lines are replaced by the conversion line with the file name wherever there is a '*'. The conversion line must send the output to stdout. For example, suppose we have a bunch of GIF files. Then we would do:

```
BASE_FILE_FORMAT PPM
INPUT
pictures.*.gif [0-10]
END_INPUT
INPUT_CONVERT giftoppm *
```

Another example: Suppose we have separate Y, U, and V files (where the U and V have already been subsampled). Then we might have:

```
BASE_FILE_FORMAT YUV
INPUT
pictures.* [0-10]
END_INPUT
INPUT_CONVERT cat *.Y *.U *.V
```

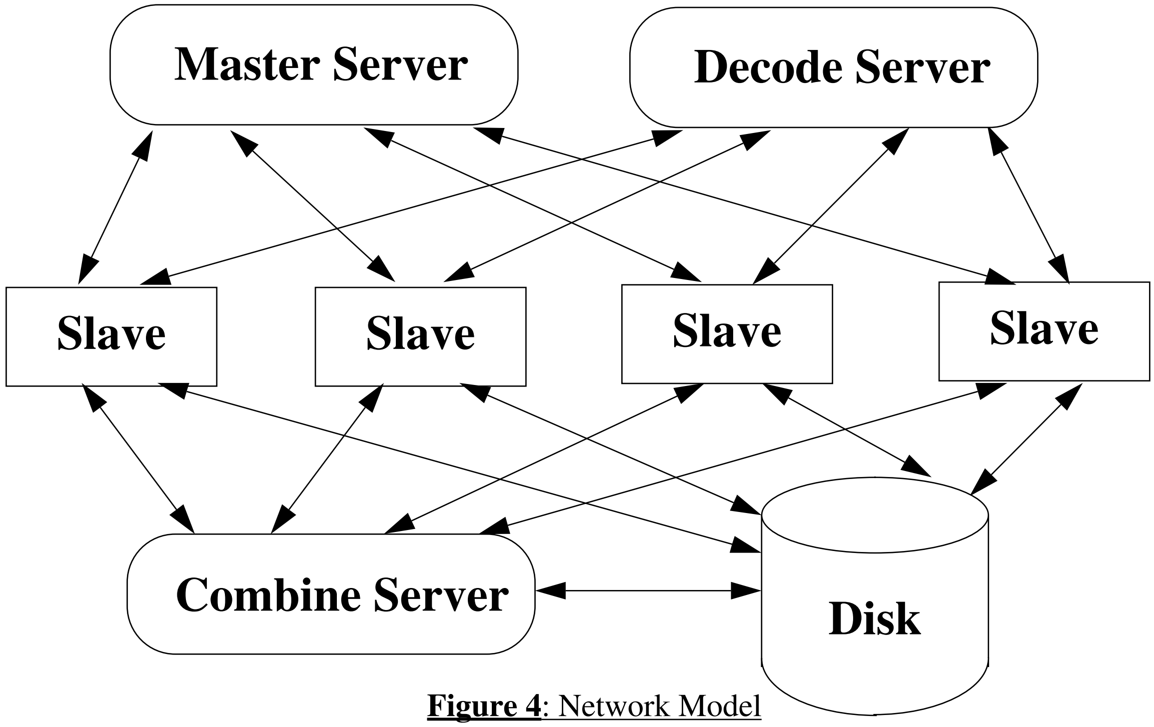

Figure 4: Network Model

As you can see, the “files” between `INPUT` and `END_INPUT` don’t have to be files at all! This can be very useful.

```
OUTPUT   /n/picasso/users/keving/encode/output/flowgard.mpg
```

2.8 Original or Decoded

The encoder can use either the original frames as reference frames, or the decoded frames. Using the decoded frames gives better playback quality, but is slower and seems to give worse compression. It also causes some complications with parallel encoding. (see the section on parallel encoding) One recommendation is to use original, and lower the q-scale if the quality is not good enough.

Table 6: Original or Decoded? (Normalized)

Reference	Compression	Speed	Quality I/P/B
Decoded	1000	1000	1000/969/919
Original	885	1373	1000/912/884

2.9 Encoding GOP’s at a Time

Instead of encoding an entire sequence, you can encode a single GOP. GOP’s can later be joined together with the encoder to form an MPEG file.

Usage:

`-gop num`

this only encodes the numbered GOP (which are numbered beginning at 0.

The output file will be the normal output filename with the suffix “.gop.<gop_num>”

GOP files can be joined at any time using the following command-line argument.

Usage:

`-combine_gops`

This causes the encoder to simply combine some GOP files into a single MPEG stream. A sequence header/ender are inserted. In this case, the parameter file need only contain the YUV_SIZE value, an output file, and perhaps a list of input GOP files. If no list of input GOP files is used, then the encoder assumes you’re using the same parameter file you used with the `-gop` option, and calculates the corresponding gop filenames itself. If this is not the case, you can specify input GOP files in the same manner as normal input files -- except instead of using INPUT_DIR, INPUT, and END_INPUT, use GOP_INPUT_DIR, GOP_INPUT, and GOP_END_INPUT. If no input GOP files are specified, then the default is to use the output file name with suffix “.gop.<gop_num>” starting from 0 as the input files.

Thus, to summarize, unless you’re mixing and matching GOP files from different sources, you can simply use the same parameter file for the `-gop` and `-combine_gops` options.

2.10 Encoding Frames at a Time

Instead of encoding an entire sequence, you can encode individual frames. These frames can later be joined together to form an MPEG file.

Usage:

-frames first_frame last_frame

This causes the encoder to encode the numbered frames in the given range, inclusive.

The output will be placed in separate files, one per frame, with the filenames being the normal output file with the suffix ".frame.<frame num>"

The frame files can later be combined as follows:

Usage:

-combine_frames

This causes the encoder to simply combine some frames into a single MPEG stream. Sequence and GOP headers are inserted appropriately. You can either use the same parameter file for -frames and -combine_frames, or you can specify frame files to combine.

The parameter file may specify input frame files in the same manner as normal input files -- except instead of using INPUT_DIR, INPUT, and END_INPUT, use FRAME_INPUT_DIR, FRAME_INPUT, and FRAME_END_INPUT. If no input frame files are specified, then the default is to use the output file name with suffix ".frame.<frame_num>" starting from 0 as the input files.

2.11 Stats and Other Options

There are several options for printing or suppressing useful information.

The encoder always prints (to stdout) parameter file information and statistics about how many I, P, and B frames there were, and information about compression and quality. You can send these statistics, in addition to the screen, to a file.

Usage:

```
-stat stat_file
```

this append the parameter file info and stats to stat_file

Normally, the statistics do not include any information about quality. This is because computing the quality takes a little more time. If you wish to have the quality included in the statistics, use the -snr command line argument.

Usage:

```
-snr
```

signal-to-noise ratio (snr) and peak snr are printed

Another set of data which can be useful is a histogram of the motion vectors. The encoder can keep track of P-frame motion vectors and forward and backward B-frame motion vectors. The output is in the form of a matrix, each entry corresponding to a motion vector in the search window. The center of the matrix represents (0,0) motion vectors.

Usage:

```
-mv_histogram
```

During normal execution, the encoder outputs two kinds of information. It prints a single line for each frame, summarizing block type and time info. It also prints, after each frame, an estimate of the remaining running time. You can modify how often or if this information is to be shown.

Usage:

```
-quiet num
-no_frame_summary
```

If `num` is negative, the time estimate is never shown; otherwise, it reports a time estimate no more often than every `num` seconds (unless the time estimate rises, which will happen near the beginning of the run). The default is `num` = 0, which means report after every frame.

If `-no_frame_summary` is given, then information about each frame is not printed.

Another nice feature is that the encoder can output the bit rate, on both a frame-to-frame scale, and also an I-frame-to-I-frame scale.

Usage:

```
-bit_rate_info rate_file
```

This puts the bit rate info into the specified file (order of info, etc.)

3. Parallel Usage

In parallel execution there are slave processes. You can have those processes run nicely if you want.

Usage:

```
-nice
```

This makes all slave processes nicely. This means that interactive users take precedence, so they don't feel like they're running in molasses. If you want to be mean to them, don't use this option. =)

3.1 Architecture Overview

See Figure 1. The slaves exist on the different slave machines which you specify (see section 3.2). The server processes all live on the machine you run the encoder on.

3.2 Specifying Slave Machines

You specify the slave machines in the parameter file. For each slave you must specify the username to use, as well as the executable mpeg_encode program. If a slave does not have NFS access, then it is REMOTE and you must also specify where the parameter file is.

Usage:

```
PARALLEL
slave_specification
END_PARALLEL
```

`slave_specification` can be either:

`machine username executable`

or

`REMOTE machine username executable param_file`

You must have an account with the given username on each machine, and you must place your machine/login in the appropriate .rhosts files.

To make it easier to run experiments with varying numbers of processors, there is a command-line argument which limits the number of slave machines.

Usage:

`-max_machines num_machines`

This means that the encoder will use no more than `num_machines` machines as slaves.

3.3 Remote Shell

To run processes on the slave machines, mpeg_encode uses the remote shell command. On most machines this is the command `rsh`. On HP machines, however, rsh is the restricted shell; on HP machines, the right command to use is `remsh`, rather than `rsh`.

Usage (parameter file):

```
RSH <rsh command>
```

3.4 Scheduling Algorithms

The encoder provides 3 different scheduling algorithms to schedule which processors get which frames.

The first scheduling algorithm simply assigns N/P frames to each processor, where N is the number of frames and P is the number of processors. This has the advantage of minimal overhead, but only works well when all the processors run at nearly the same speed. Also, since most processors will finish at about the same time, you will have to wait at the end while the Combine Server gathers all the frame files together.

Usage:

PARALLEL_PERFECT

The second scheduling algorithm first assigns S frames to each processor. When a processor is finished, it is assigned T seconds of work (the scheduler estimates this based on previous performance). S should be at least 3, preferably at least 5 or 6, to insure a good estimate of each processor's speed.

Usage:

PARALLEL_TEST_FRAMES S
PARALLEL_TIME_CHUNKS T

The third scheduling algorithm is probably the best. It also first assigns S frames to each processor. Subsequently, however, whenever a processor finishes, it is assigned enough work to keep it busy until almost everything is done. Effectively, a processor is assigned many frames, and then fewer and fewer frames as more work gets done. This insures good load balancing, while limiting scheduling overhead.

Usage:

PARALLEL_TEST_FRAMES S
PARALLEL_CHUNK_TAPER

4. Performance

Here is a look at sequential performance.

Table 7: Machine Comparison

Machine	MPS[a]
HP 9000/755	280
DEC 3000/400	247
HP 9000/750	191
Sparc 10	104
DEC 5000	68

a. Macroblocks per second; a 320x240 pixel image is 300 macroblocks per frame.

Parallel performance is dependent not only on processor performance, but network performance. If you are using a 10 Mb/s Ethernet, don't expect to get better than 4 or 5 frames per second -- no matter how fast your processors are.

Parallel performance is also greatly dependent on how big the input files are (YUV is better than PPM, and JPEG is better than both), and how big the output files are (better compression will lead to less I/O).

5. Other Options

See Figure 1. The slaves exist on the different slave machines which you specify (see section 3.2). The server processes all live on the machine you run the encoder on.

5.1 Custom Quantization Tables

You can specify your own custom quantization tables. Currently, you can only do this once per MPEG file. You can specify both Intra- and Non-intra quantization tables. If you don't specify them, then the default tables are used.

Usage:

```
NIQTABLE
table row 1
table row 2
...
table row 8
```

This specifies the Intra quantization table. Each `table row` is simply 8 integers, separated by tabs and/or spaces.

Usage:

```
IQTABLE
table row 1
table row 2
...
table row 8
```

This specifies the non-intra quantization table.

5.2 Aspect Ratio

You can specify the aspect ratio. I'm not sure what this is used for.

Usage:

`ASPECT_RATIO float`

`float` is one of {1.0, 0.6735, 0.7031, 0.7615, 0.8055, 0.8437, 0.8935, 0.9157, 0.9815, 1.0255, 1.0695, 1.0950, 1.1575, 1.2015}.

5.3 Frame Rate

You can specify the frame rate. This is used by some playback systems to gauge the playback rate.

Usage:

`FRAME_RATE float`

`float` is one of {23.976, 24, 25, 29.97, 30, 50, 59.94, 60}.

6. Other Tools

The misc/ directory contains several useful tools.

6.1 ppmtoyuv

Usage:

`ppmtoyuv < input.ppm > output.yuv`

This takes as input a ppm file and outputs a subsampled yuv file suitable for the encoder.

6.2 jmovie2jpeg

Usage:

`jmovie2jpeg infile outfile start-frame end-frame`

`infile` is a version 2 Parallax J_Movie

`outfile` is a base file name for the output files

`start-frame` and `end-frame` are the starting and ending frame numbers

This takes as input a J_Movie and creates separate JFIF compatible JPEG files with the names base<num>.jpg, where base is outfile, and <num> are the frame numbers.

jmovie2jpeg was written by Jim Boucher (jboucher@flash.bu.edu).

6.3 movieToVid

Usage:

`movieToVid movieFile dataDir indexDir srcHostName`

This program is used to convert a Parallax J Movie into a ".vid" file, which is video only. vid files are used by some of the programs described later.

See the README file in misc/mtv/ for more details on usage.

movieToVid was written by Brian Smith (bsmith@cs.berkeley.edu)

6.4 yuvtojpeg

Usage:

`yuvtojpeg infile outfile`

This takes as input an encoder yuv file and outputs a jpeg file. It uses cjpeg to do the compression.

6.5 blockrun

Usage:

`blockrun command num_args firstnum lastnum skip arg1 ... argn`

This runs the given command (which has `num_args` args), with the args `arg1 ... argn`, where any '=' character is replaced by a number from `firstnum` to `lastnum` skipping by `skip`. For example:

```
blockrun yuvtojpeg 2 13 19 3 flow=.yuv flow=.jpg
```

will run:

```
yuvtojpeg flow13.yuv flow13.jpg
yuvtojpeg flow16.yuv flow16.jpg
yuvtojpeg flow19.yuv flow19.jpg
```

6.6 vidtoppm

Usage:

```
vidtoppm filename width height start end outbase [quality]
```

This takes as input a .vid file of given `height` and `width`, and turns them into a bunch of ppm files named `outbase`.N, where N is a number from `start` to `end`.

6.7 vidtojpeg

Usage:

```
vidtojpeg filename width height start end outbase [quality]
```

This is the same as vidtoppm, except it outputs JPEG files instead of PPM files.

6.8 vidtoyuv

Usage:

```
vidtoyuv filename width height start nth outbase [quality]
```

This takes as input a .vid file of given `height` and `width`, and turns them into a bunch of yuv files named `outbase`.N, where N is a number from `start` to `end`, skipping by `nth`.

7. Frequently Asked Questions

7.1 Questions

1. How do I encode a sequence that can be played by the Xing player?
2. I'm using the Parallax XVideo card to digitize; how do I MPEG-encode the resulting data?
3. How do I convert the MPEG-encoder YUV files into PPM files?

7.2 Answers

1. The XING player samples video at 160x120 and expands to output 320x240. This is where their speed comes from. The player cannot buffer a 320x240 and thus had data overruns. The xing player would 'theoretically' handle 160x120 I frames.

Thus, to encode, use PATTERN I and 160x120 frames. (jboucher@flash.bu.edu)

2. There will be better tools to do this later, but currently the easiest thing to do is use the jmovie2jpeg utility in the misc/ directory.

3. Stanford's CVv1.2.2.tar.Z includes cyuv2ppm.c. Which after you split the Y, U, and V components out, works fine. (curly@hsn.cftnet.com)