EE669 2015 Spring

PROJECT 3: Video Coding using MPEG-2

Project Report

Name: Shanglin Yang ID 3795329308

Email: shangliy@usc.edu

Table of Contents

Problem 1: Written questions

A: Difference between MPEG1 and MPEG2

MPEG1 and MPEG2 are both standards for the generic coding of moving pictures and associated audio information. These standards describe the combined lossy compression of audio and video procedure which allows the storage and transmission of moving pictures with audio.

The compression standard for VHS quality digital video with a CD audio down to 1.5 Megabits per second is MPEG-1. In MPEG-1, the compression ratio of video without losing too quality is 26:1 and the ratio or audio is 6:1. This type of compression makes it viable for digital audio and TV broadcasting as well as the creation of video CDs. As a consequence, this lossy audio and video format has become hugely popular due to its wide compatibility. Various products and applications use the MPEG-1 standard especially the audio format it introduced, the extremely popular MP3.

There are mainly three difference between MPEG1 and MPEG 2 ;

1: MPEG1 only provides Progressive Scan; while MPEG 2 provides Progressive and Interlaced Scan;

2: The Out-stream of MPEG1 contains only Program stream while MPEG2 contains Program Stream and Transport Stream;

3:The audio compression of MPEG1 is limited in mono and stereo channel ; while the MPEG2 involves mono, stereo and 5.1 channel.

B: FFMPEG memory management

FFmpeg apply for memory for the current block and related block rather than the whole stream. After encoding, then the ffmpeg free and reallocate the memory (free the previous and the require the memory) thus, it will not take much memory.

There are three key function in the **transcode_init(void)** function in the ffmpeg.c:

void* av_mallocz(size_t size)av_malloc_attrib1

Allocate a block of size bytes with alignment suitable for all memory accesses (including vectors if available on the CPU) and zero all the bytes of the block.

Void* av_realloc(void*ptr, size_t size)

Allocate or reallocate a block of memory.

Void av_free (void *ptr)

Free a memory block which has been allocated with av_malloc(z)() or av_realloc().

C: Ffmpeg Program Flow

1: The main flow in the ffmpeg.c

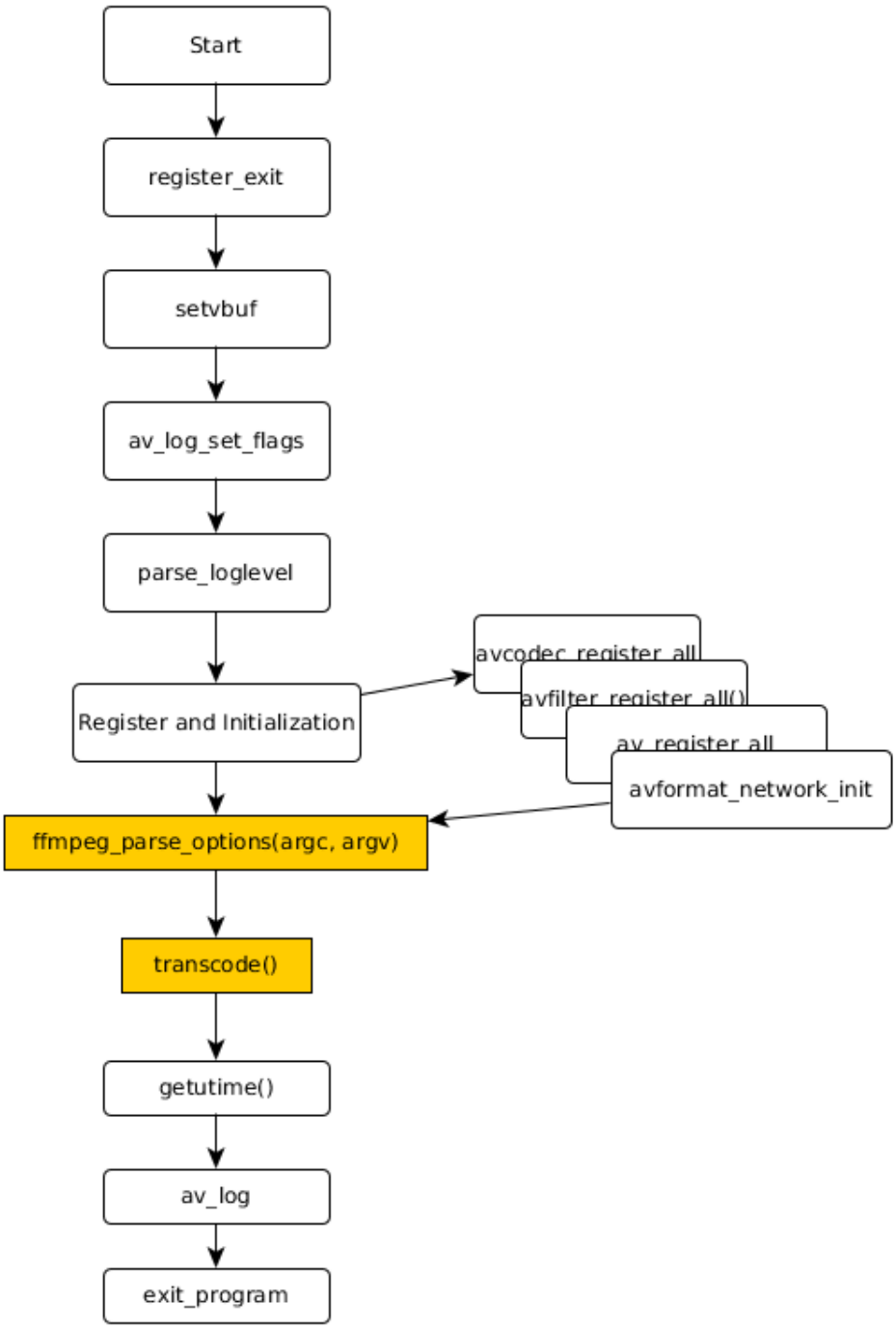


Figure 1 main flow in the ffmpeg.c

2: The main loop in the train

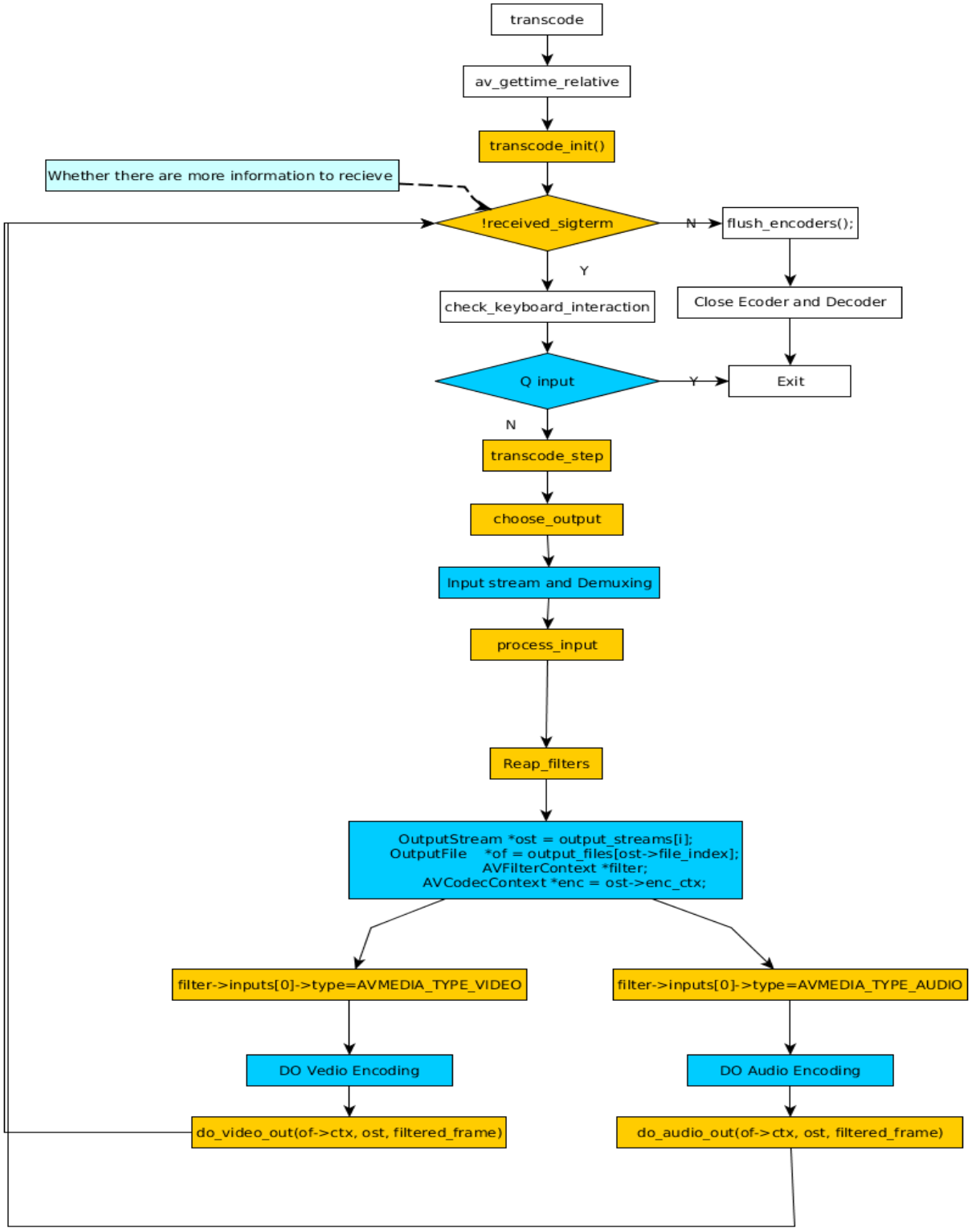


Figure 2 Main loop in the train

3: The main process for mpeg2 encode

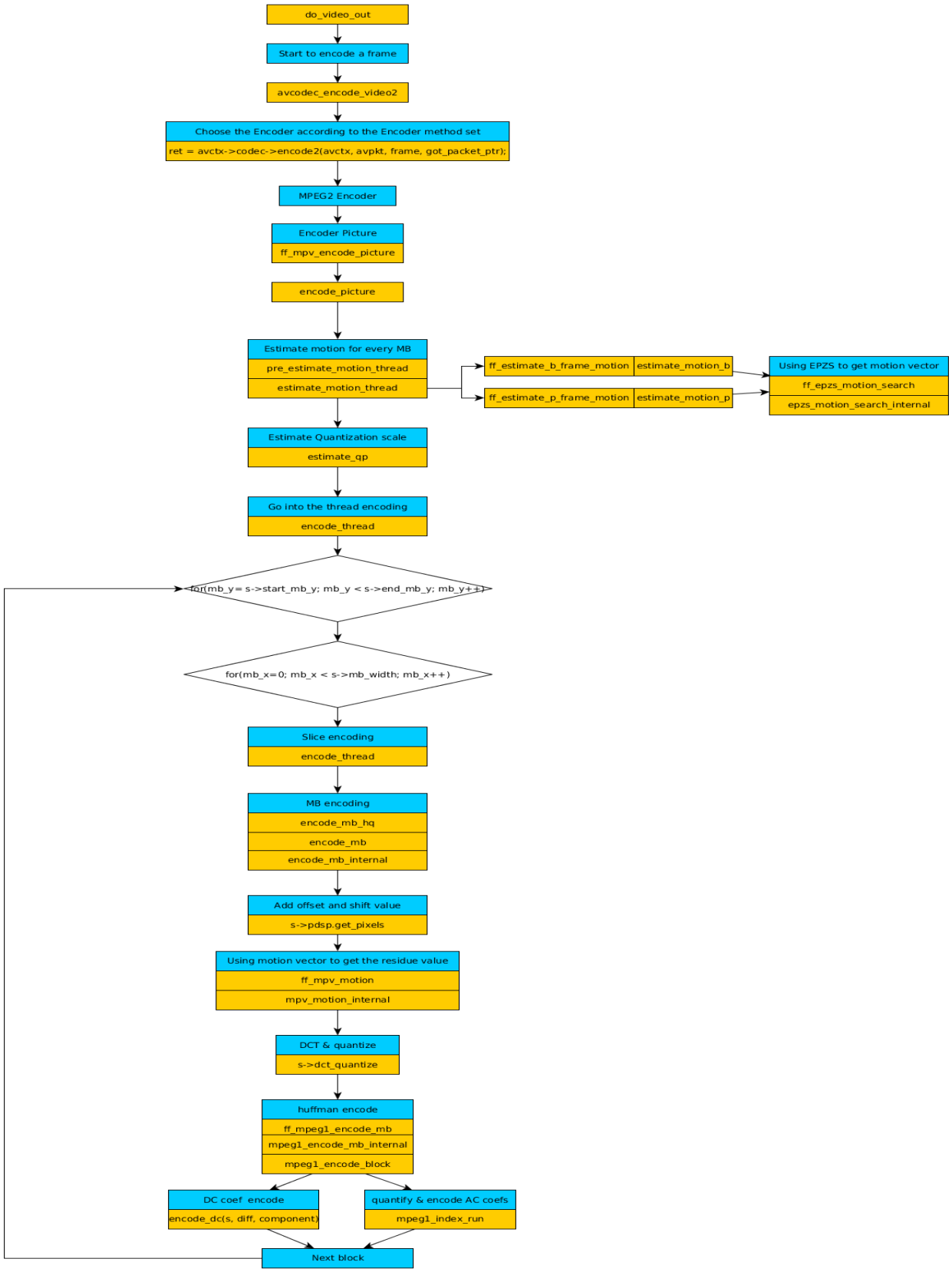


Figure 3 Main process for mpeg2 encoder

4: The main function in the whole process

1': void parse_options(void *optctx, int argc, char **argv, const OptionDef *options,void (*parse_arg_function)(void *, const char*))

Function->Analysis and Parse the command line arguments.

Parameter->

***optctx, argc,argv :** Input parameter (commands from user)

void (*parse_arg_function): param parse_arg_function Name of the function called to process every argument without a leading option name flag.Parse one given option. Return on success 1 if arg was consumed, 0 otherwise; negative number on error.

2':static int transcode(void)

Function->The main loop of the file converter

3':static int transcode_init(void)

Function->initialization the transcode process

4':static int transcode_step(void)

Function-> Run a single step of transcoding.

return 0 for success, <0 for error

5':static OutputStream *choose_output(void)

Function-> Select the output stream to process.

return selected output stream, or NULL if none available

6': static int process_input(int file_index)

Function-> Get input the stream data and demuxing the video and audio

Input->The file name;

Output->pointer pointing the input data stream

7':static int reap_filters(void)

Function->Get and encode new output from any of the filtergraphs, without causing

activity.

Output->return 0 for success, <0 for severe errors

8':static void do_video_out(AVFormatContext *s,

OutputStream *ost,

AVFrame *next_picture)

Function->Encode a frame of video;

Takes input raw video data from frame and writes the next output packet, if available, to avpkt. The output packet does not necessarily contain data for the most recent frame, as encoders can delay and reorder input frames internally as needed.

Input->*s: main constructure for encoding;

OST:POINT TO THE OUTSTREAM

AV-FRAME:POINT TO THE NEXT PICTURE DATA;

9':int avcodec_encode_video2(AVCodecContext *avctx, AVPacket *avpkt,

const AVFrame *frame, int *got_packet_ptr);

Function-> Encode a frame of video and choose the encoder according to the method to use;

Input-> avctx: codec context

avpkt: output AVPacket.

frame:AVFrame containing the raw video data to be encoded.

got_packet_ptr:This field is set to 1 by libavcodec if the

output packet is non-empty, and to 0 if it is

empty. If the function returns an error, the

packet can be assumed to be invalid, and the

value of got_packet_ptr is undefined and should

not be used.

Output-> 0 on success, negative error code on failur

10': static av_always_inline int epzs_motion_search_internal(MpegEncContext * s, int *mx_ptr, int *my_ptr,int P[10][2], int src_index, int ref_index, int16_t (*last_mv)[2],int ref_mv_scale, int flags, int size, int h)

Function-> Do the motion vector estimation using epzs method;

Input-> * s: codec context.

*mx_ptr: point to the motion_x

*my_ptr: point to the motion_y

P[10][2]:a list of candidate mvs to check before starting the

iterative search. If one of the candidates is close to the optimal mv, then it takes fewer iterations. And it increases the chance that we find the optimal mv.

ref_index: Reference picture index.

Output-> motion vector

11':static av_always_inline void encode_mb_internal(MpegEncContext *s,

int motion_x, int motion_y,

int mb_block_height,

int mb_block_width,

int mb_block_count)

Function-> Do the block encoding;

Input-> * s: codec context.

motion_x: motion_x

motion_y: motion_y

mb_block_height: block_height

mb_block_width: block_width

mb_block_count:the number of blocks

12': s->dct_quantize(s, s->block[i], i, s->qscale, &overflow);

Function-> Do the DCT and Quantization;

Input-> * s: codec context.

s->block[i]: point to current block

i: current block index

s->qscale:Quantization scale

overflow:The flag of overflow;

13': encode_dc(s, diff, component)

Function->Encode DC coefficients

Input-> * s: codec context.

diff: The difference between current DC and reference DC

component: Index for the DC value.

D: performance optimization

1: Using much bitwise operation to speed up the calculation;

2: using more than one threads to do the Decoder and Encoder simultaneously;

3: Using pointer and construction to transfer data avoding multi-times of data read and store.

Problem 2: Encoder Configuration and Motion Estimation

2.1 Abstract and Motivation

FFmpeg is a free software project that produces libraries and programs for handling multimedia data. FFmpeg includes libavcodec, an audio/videocodec library used by several other projects, libavformat, an audio/video container mux and demux library, and the ffmpeg command line program for transcoding multimedia files. FFmpeg is published under the GNU Lesser General Public License 2.1+ or GNU General Public License 2+ (depending on which options are enabled).

MPEG1 and MPEG2 are both standards for the generic coding of moving pictures and associated audio information. These standards describe the combined lossy compression of audio and video procedure which allows the storage and transmission of moving pictures with audio.

There are many parameters along with the encoder and decoder process which leads to different quality, bits of file and other value of the result. Studying the process not only help us to have a deep understand of the MPEG process and also improve the ability of solving the multimedia problem to meet different requirements and situations.

Besides, the motion estimation is a very part in the mpeg coding. Good motion estimation leads to the smaller residue and leads to smaller bit/rate for same quality. Meanwhile, the motion estimation is also involved with the memory using and calculation complexity which are important for the practical application.

2.2 Aproach and Procedures

2.2.1 Encoder and decoder using Ffmpeg

Encoder-> Using command:

ffmpeg -f rawvideo -vcodec rawvideo -s 352x288 -r 25 -threads 2 -pix_fmt yuv420p -i foreman_cif.yuv -c:v mpeg2video -me_method epzs -r 25 -g 12 -b:v 250k -maxrate 250k -minrate 250k -psnr -benchmark -vstats_file foreman_state.txt foreman.mp4

ffmpeg -f rawvideo -vcodec rawvideo -s 352x288 -r 25 -threads 2 -pix_fmt yuv420p -i akiyo_cif.yuv -c:v mpeg2video -me_method epzs -r 25 -g 12 -b:v 250k -maxrate 250k -minrate 250k -psnr -benchmark -vstats_file akiyo_state.txt akiyo.mp4

-f rawvideo: Input data property;

-vcodec rawvideo: Input data coding method;

-s 352*288: Input data size;

-r 25: Input data frame rate equal to 25;

-threads 2: Set thread detection to 2;

-pix_fmt yuv420p: Input data is coded with 4:2:0 format;

-i foreman_cif.yuv: Input file name (-i akiyo_cif.yuv for akiyo);

-c:v mpeg2video: Encoder method using Mpeg2;

-me_method epzs: Set motion estimation method to EPZS

-r 25: Output data frame rate equal to 25;

-g 12: Set GOP size to 12;

-b:v 250k -maxrate 250k -minrate 250k: Set the output bitrate to 250kbits/s;

-psnr: Output PSNR;

-benchmark: Output time used;

-vstats_file foreman_state.txt: Output information to file foreman_state.txt;

foreman.mp4: Output file name;

Decoder-> Using command:

ffmpeg -i foreman.mp4 -c:v rawvideo foreman_cif_dec.yuv

2.2.2 B-frame and Gop size effect

(1) Inserting B frame

Using command -bf n to insert n B-frames between adjacent I/P or P/P-frames.

ffmpeg -f rawvideo -vcodec rawvideo -s 352x288 -r 25 -threads 2 -pix_fmt yuv420p -i foreman_cif.yuv -c:v mpeg2video -me_method epzs -r 25 -g 12 -b:v 250k -maxrate 250k -minrate 250k -bf 1 -psnr -benchmark -vstats_file foreman_state.txt foreman.mp4

(2) Change GOP size

change the -g 8 -g 15 -g 30

(3) Change the Bit/Rates

change -b:v 250k -maxrate 250k -minrate 250k

2.2.3 Motion Estimation

(1) Extract the motion vector

The motion estimation function are in the file

" motion_est.c " and "motion_est_template.c ".

The estimation flow is seen below:

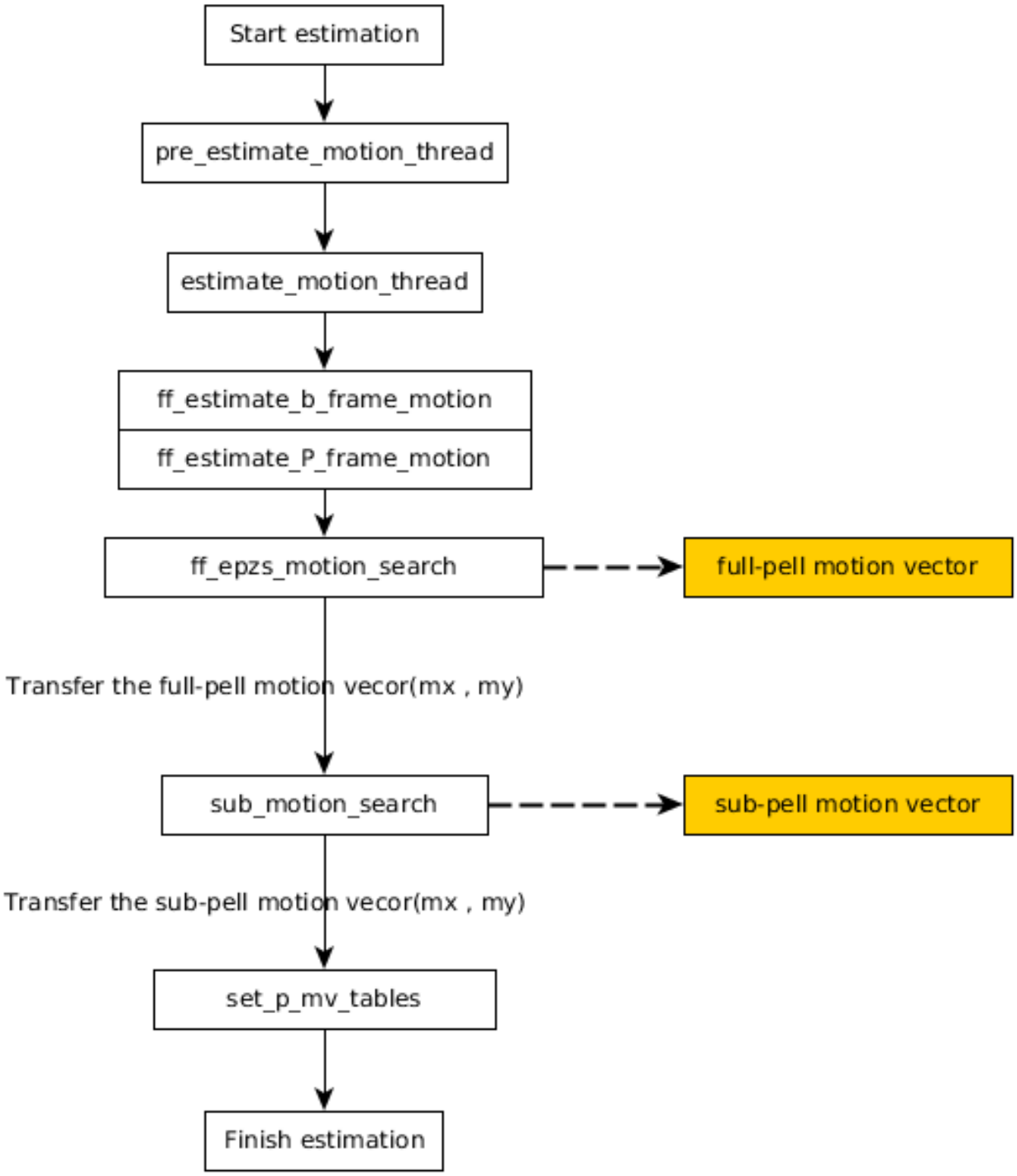


Figure 4 estimation flow

Find there are two kind of the motion vector:

Full-pell: Based on the original pixel;

Sub-pell: Based on the new (half pell inserted) pixel;

we can out put the motion vector at the specific place as shown above.

Use the coed in motion_est.c to extract the motion vector the files :

```
fopen_x=fopen("mv_b_x_full.txt","a+");  // Open the output files for motionx
fopen_y=fopen("mv_b_y_full.txt","a+");  // Open the output files for motiony
 if (s->pict_type==AV_PICTURE_TYPE_P) //Get when get P-frame
  {
        fprintf(fopen_x,"%d\n",mx);  //Outut motion_x
       fprintf(fopen_y,"%d\n",my);  //Outut motion_y
  }

fclose(fopen_x);
fclose(fopen_y);
```

(2) Plot 3D histogram

Use Matlab import Data and Use the command below to get the 3D histogram.

```
figure(1)
x=a_300_full_x;  %import data
y=a_300_full_y;
max_p=max([max(x);max(y)]); %Find the max and min
min_p=min([min(x);min(y)]);
[X,Y] = meshgrid(min_p:1:max_p); %Grid the pixel for plot
cnt1 = hist3([x,y], {min_p:1:max_p min_p:1:max_p}); % Calculate the Histogram
mesh(X,Y,cnt1') %Plot the 3d histogram
xlabel('x-motion'); ylabel('y-motion');
```

(3) Implemention motion estimation method is EPZS

a: The EPZS method is based on the previous algorithm like PMVFAST, it's key idea is to introduce new motion prediction candidate in the subset to do the prediction.

b: It's main process flowchare is seen below.

Figure 5 Implemention motion estimation method is EPZS flow

c :Main points

(1) 3 Subsets are used in in the estimation process:

Subset A: median predictor; Subset B: motion vector of three adjacent blocks (left, top, top-right) and (0,0); Subset C: accelerator motion vector (considering blocks could be accelerating)+4 motion vectors of the blocks around the collocated block in the previous frame (for the collocated block have high velocity so it may be correlated to more to the previous frame than the current fram).The C is more likely to be the predictor

(2) Thresholding parameters were adaptively calculated by considering all these candidates including the previous frame. In general(a_k and b_k are fixed parameters):

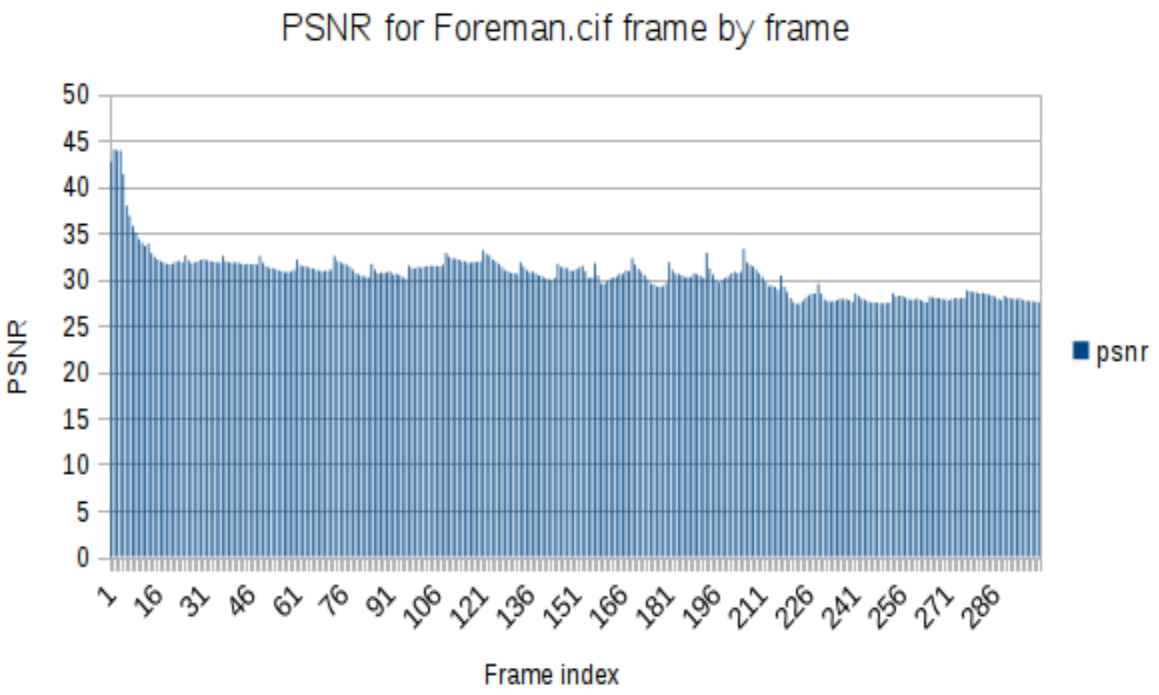
$$T_k = a_k \times \min(MSAD_1, MSAD_2, \ldots, MSAD_n) + b_k$$

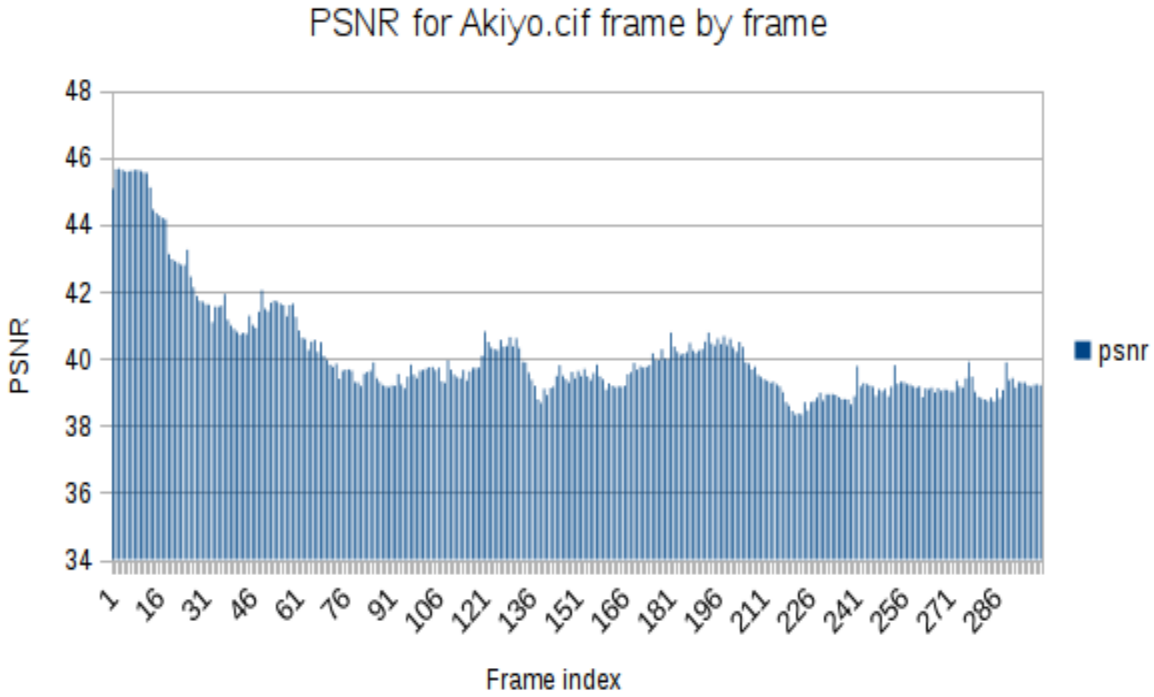
(3) For search pattern, considering we get good prediction of the start ponts ,we can use smaller diamond zone shapes or 8-point square/ circular pattern to save the window size.

(4)EPZS can also be implemented in an N-dimensional framework using blocks and search pattern with high dimension.

2.3 Results

2.3.1 Encoder and decoder using Ffmpeg





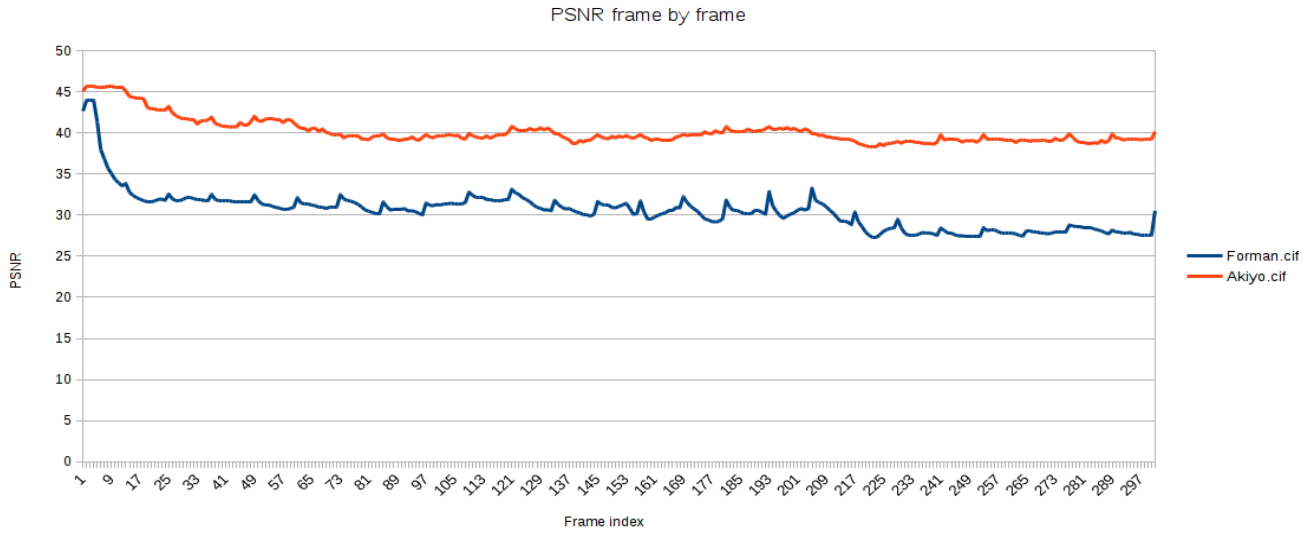


Figure 6 PSNR for two stream frame by frame

Table 1 Encoding results for two stream

	Final LPSNR				Average PSNR	Average bit rates (kBits/s)	total encoding time(s)
	Y	U	V	MB(*)			
Foreman.cif	30.02	37.86	38.52	31.46	30.52	306.9	0.831
Akiyo.cif	39.93	43.18	44.87	40.9	40.16	314.6	0.765

2.3.2 B-frame and Gop effect

(1) Inserting B frame

Table 2 Encoding results for Foreman with different B-frame number

Foreman.cif	Final PSNR				Average bit rates (kBits/s)	total encoding time(s)
	Y	U	V	MB(*)		
B=0	30.02	37.86	38.52	31.46	306.9	0.831
B=1	29.97	37.88	38.57	31.42	305	1.096
B=2	29.53	37.73	38.36	31	304.6	1.578
B=3	29.38	37.63	38.24	30.85	321.1	1.782

Table3　Encoding results for Akiyo with different B-frame number

Akiyo.cif	Final PSNR				Average bit rates (kBits/s)	total encoding time(s)
	Y	U	V	MB(*)		
B=0	39.93	43.18	44.87	40.9	314.6	0.765
B=1	40.31	43.36	45.19	41.27	314.8	1.148
B=2	40.42	43.6	45.25	41.38	310.5	1.302
B=3	40.28	43.47	45.25	41.25	306.4	1.387

(2) Change GOP size

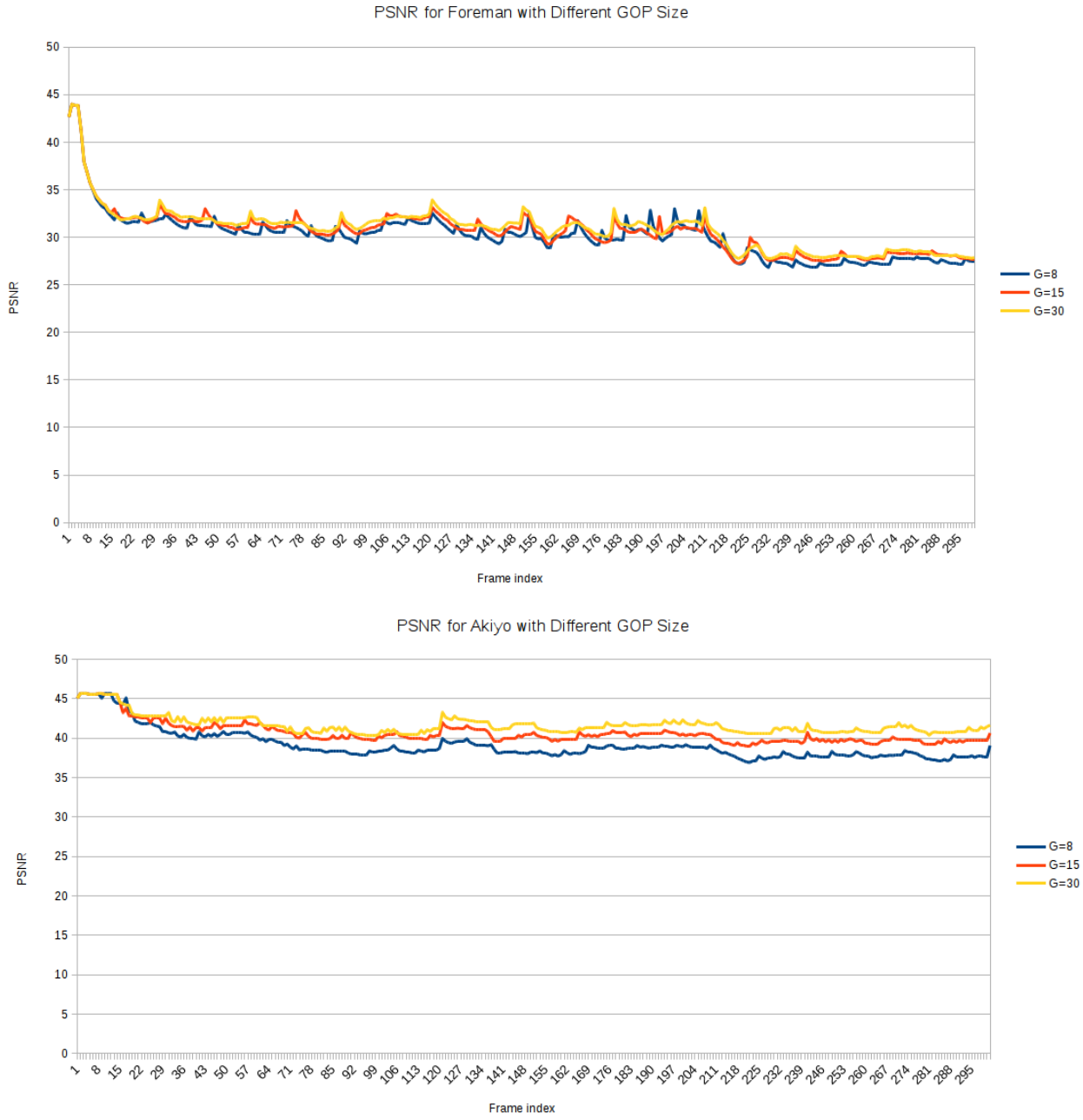


Figure 7　PSNR for two stream using different GOP size frame by frame

Table 4 Encoding results for two stream using different GOP size

	Y	U	V	MB(*)	Average PSNR	Average bit rates (kBits/s)	total encoding time(s)
G=8	29.64	37.83	38.48	31.1	30.54	317.1	0.907
G=12	30.02	37.86	38.52	31.46	30.52	306.9	0.934
G=15	30.12	37.81	38.47	31.55	30.90	304.3	0.932
G=30	30.41	37.57	38.31	31.8	30.92	299.1	0.973

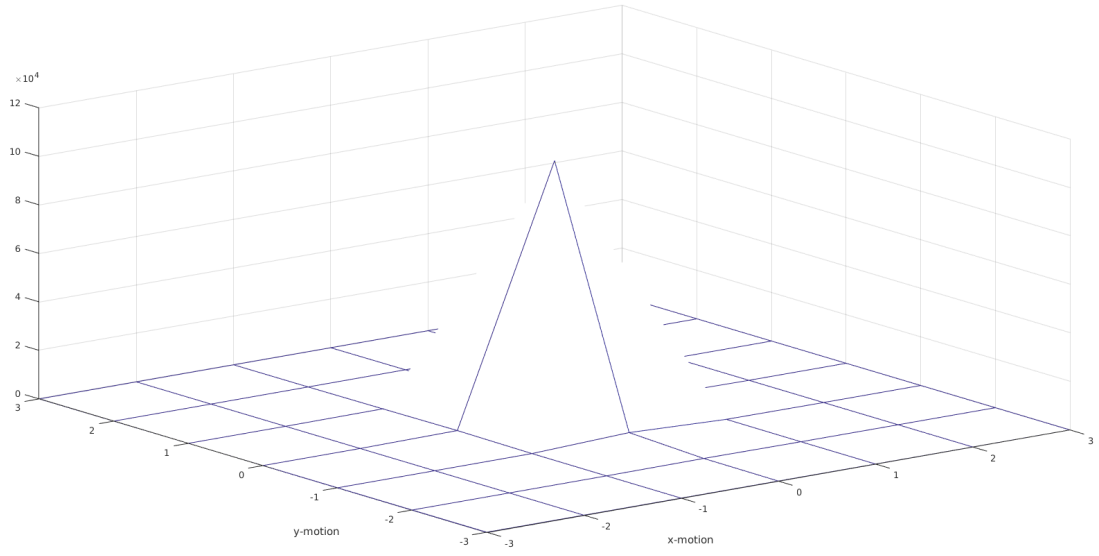
Akiyo.cif	Fina LPSNR				Average PSNR	Average bit rates (kBits/s)	total encoding time(s)
	Y	U	V	MB(*)			
G=8	39.75	42.28	44.11	39.78	39.06	325.2	0.743
G=12	39.93	43.18	44.87	40.9	40.16	314.6	0.765
G=15	40.45	43.59	45.21	41.41	40.64	309.8	0.788
G=30	41.51	44.47	45.79	42.41	41.64	301.2	0.822

(3) Change the Bit/Rates

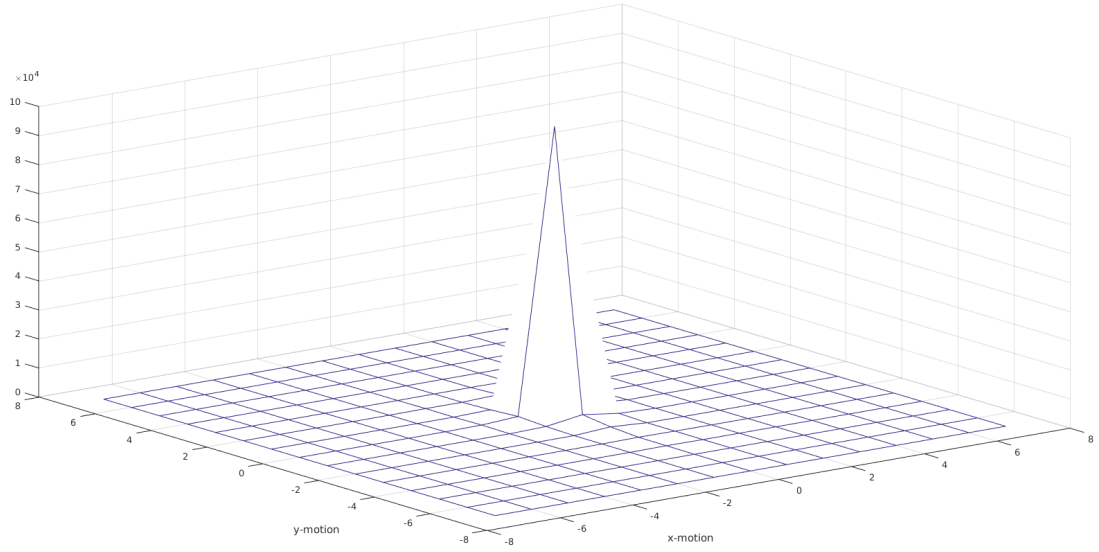
Table 5 Encoding results for two stream using different Bitrates and B-frame

			Final LPSNR				Average bit rates (kBits/s)	total encoding time(s)
	Bitrates(kbps)	B-FRME	Y	U	V	MB(*)		
Foreman.cif	100	**B=0**	**28.66**	**37.13**	**37.58**	**30.13**	**230.8**	**0.96**
		B=1	28.98	37.38	37.9	30.45	262	1.509
		B=2	29.04	37.34	37.85	30.51	282.8	1.785
		B=3	29.08	37.45	37.96	30.55	301.1	2.077
	200	**B=0**	**29.32**	**37.46**	**38.04**	**30.78**	**259.3**	**0.855**
		B=1	29.23	37.53	38.08	30.7	269.3	1.25
		B=2	29.26	37.5	38.05	30.72	285.1	1.682
		B=3	29.27	37.57	38.16	30.74	320	1.769
	300	**B=0**	**30.89**	**38.21**	**38.94**	**32.39**	**354.3**	**0.911**
		B=1	31.1	38.41	39.19	32.5	347.9	1.426
		B=2	30.76	38.19	38.93	32.18	340.8	1.598
		B=3	30.19	38.04	38.66	31.63	337.5	1.747
Akiyo.cif	100	**B=0**	**36.17**	**40.02**	**42.17**	**37.26**	**165.6**	**0.786**
		B=1	35.41	39.2	41.77	36.52	170.3	1.217
		B=2	35.12	38.97	41.56	36.23	169.2	1.321
		B=3	34.38	38.15	41.02	35.5	175.9	1.46
	200	**B=0**	**39.42**	**42.84**	**44.51**	**40.42**	**263**	**0.755**
		B=1	39.54	42.87	44.65	40.53	263.3	1.17
		B=2	39.68	43.03	44.8	40.68	258.9	1.328
		B=3	38.93	42.34	44.26	39.95	261.4	1.42
	300	**B=0**	**41.32**	**44.31**	**45.8**	**42.23**	**358.5**	**0.817**
		B=1	41.78	44.7	46.19	42.69	358	1.172
		B=2	41.99	44.96	46.4	42.9	353.2	1.215
		B=3	41.42	44.5	46.1	42.36	350.4	1.422

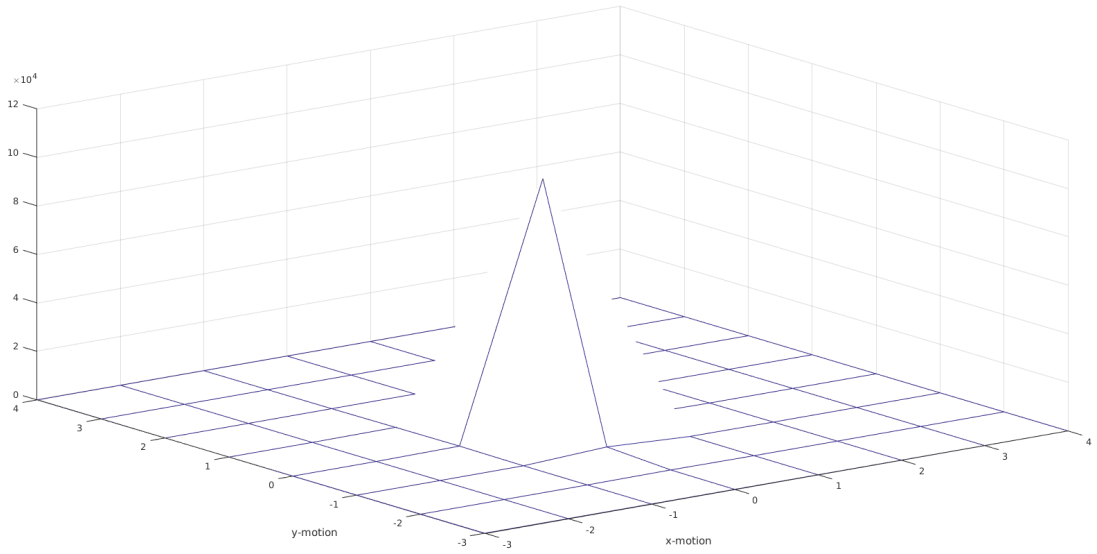
2.3.3 Motion Estimation



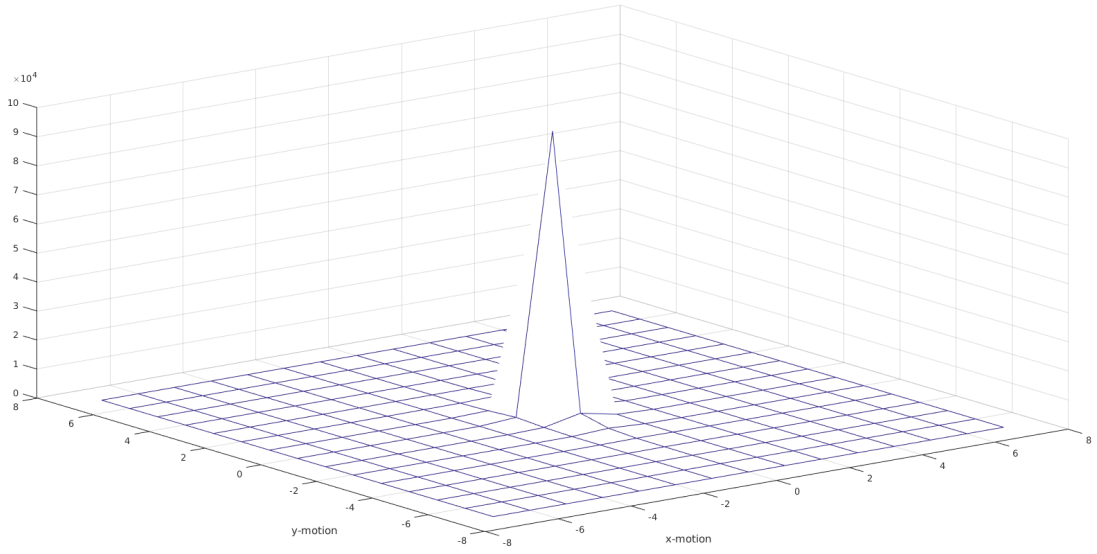
akiyo_bite=100k_fullpixel_MV histogram



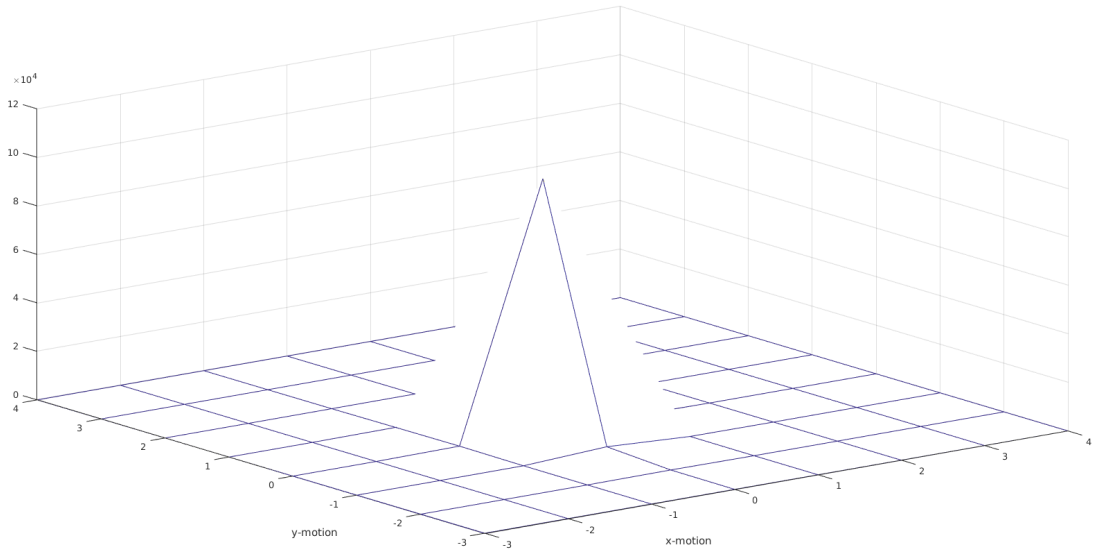
akiyo_bite=100k_sublpixel_MV histogram



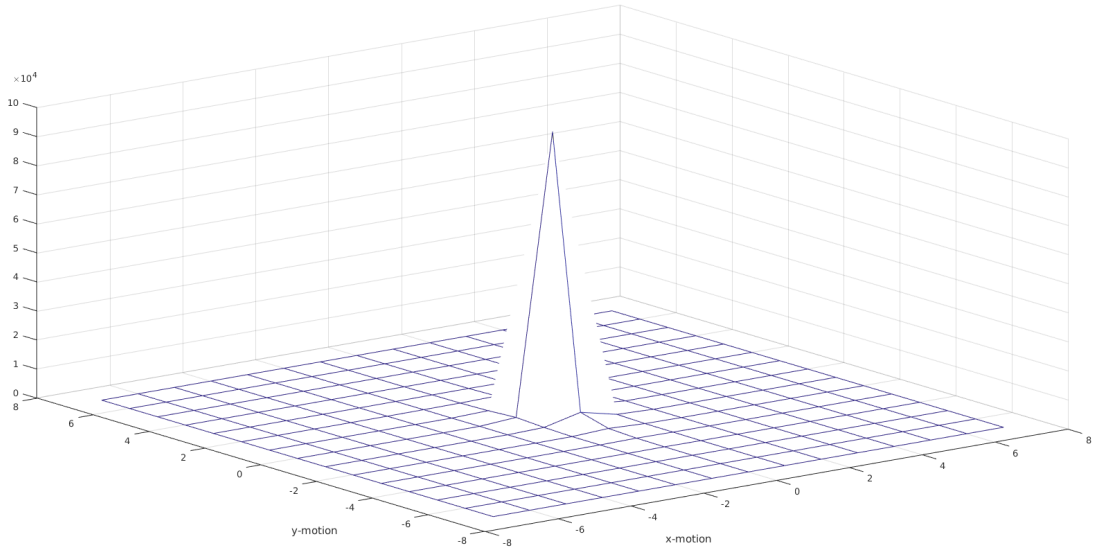
akiyo_bite=200k_fullpixel_MV histogram



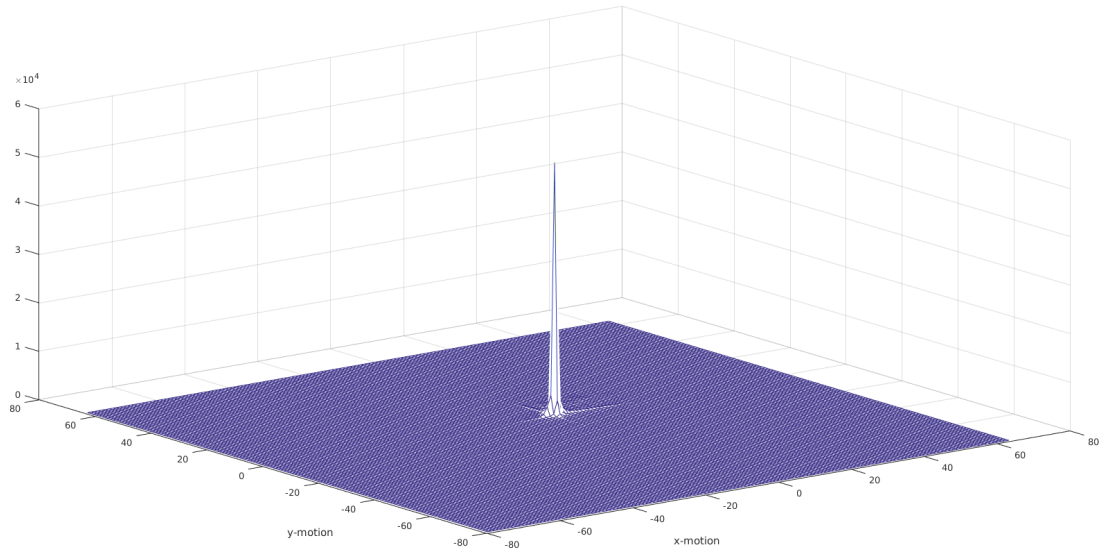
akiyo_bite=200k_subpixel_MV histogram



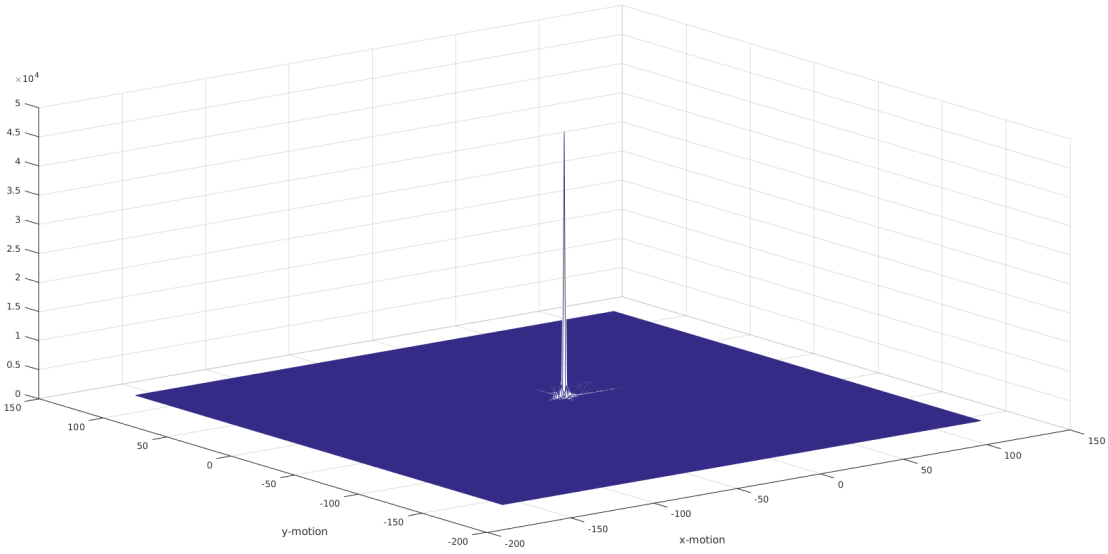
akiyo_bite=300k_fullpixel_MV histogram



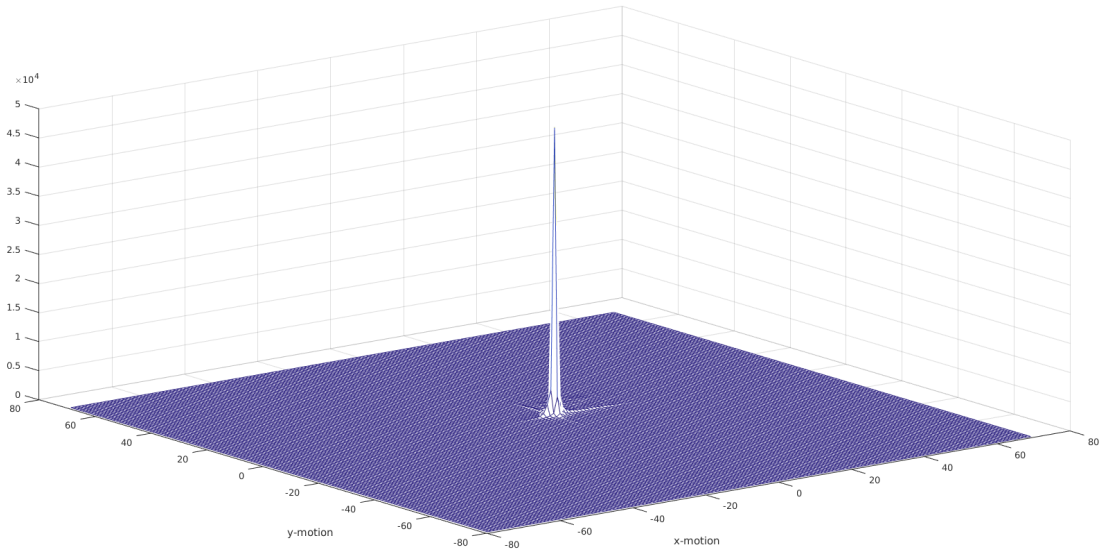
akiyo_bite=300k_subpixel_MV histogram



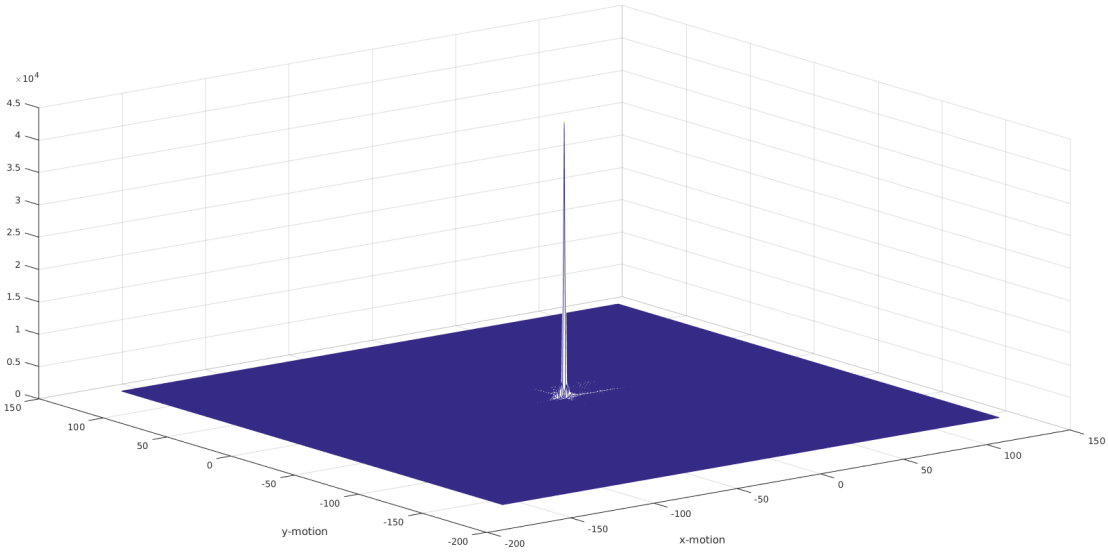
foreman_bite=100k_fullpixel_MV histogram



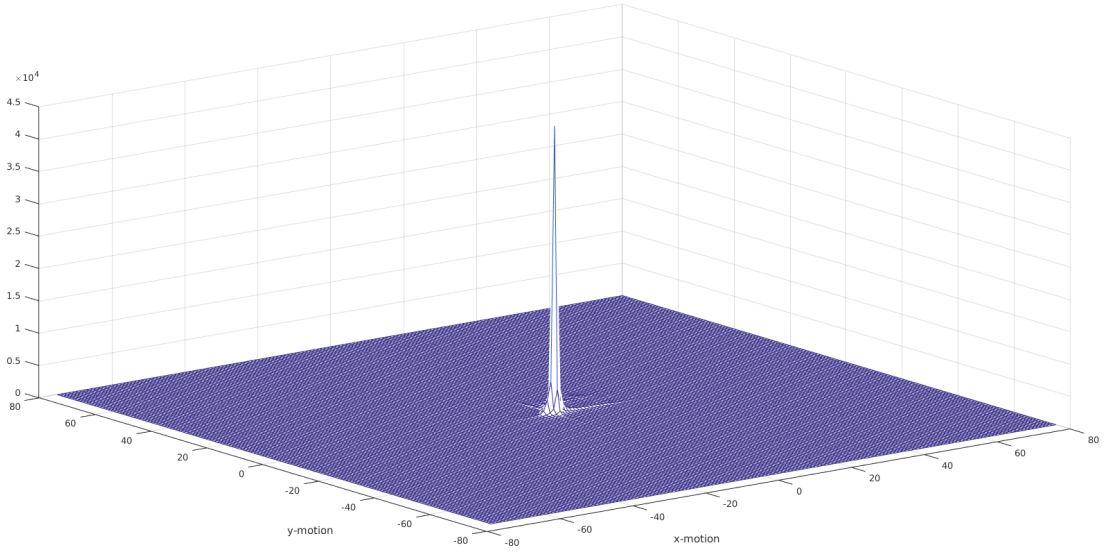
foreman_bite=100k_subpixel_MV histogram



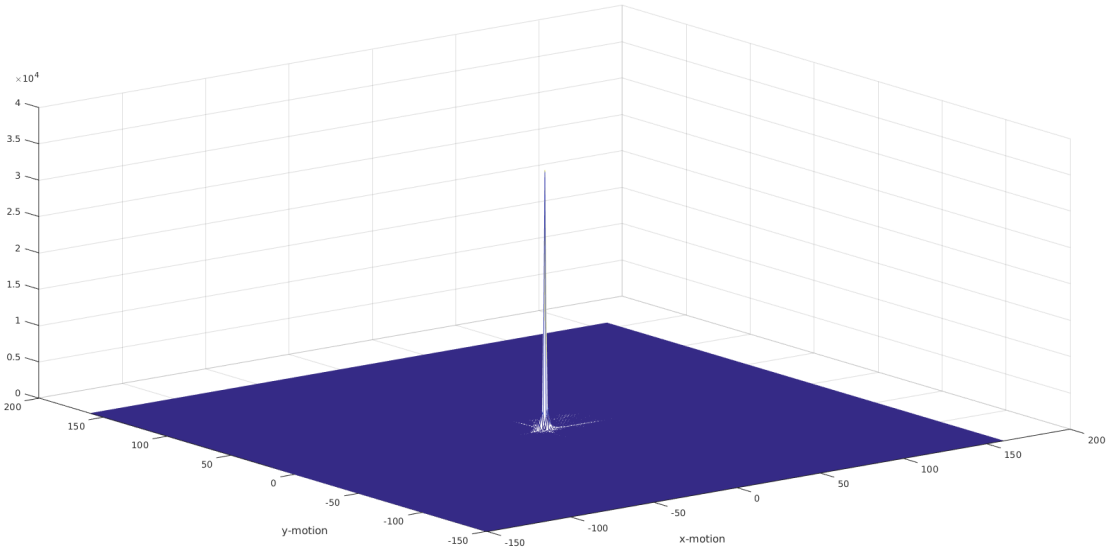
foreman_bite=200k_fullpixel_MV histogram



foreman_bite=200k_subpixel_MV histogram



foreman_bite=300k_fullpixel_MV histogram



foreman_bite=300k_subpixel_MV histogram

Figure 8　3-D histogram for two stream with different bitrates

2.4 Discusion

2.4.1 Encoder and decoder using Ffmpeg

(1)Comparision between Akiyo and Foreman

The PSNR of Akiyo is higher than Foreman and the bitrates of Akiyo is also higher than Foreman.

The reason for that is that the Akiyo is relatively stationary. On the one hand, the prediction of motion vector could be more accurate to save bits, on the other hand, the size of motion vector should also be smaller. All contributes to using smaller bitrates to get better results.

(2)Comparision between the original and resulting videos

Playing the original and decoded video, could find the quality is low for the size of file is reduced largely. From the comparsion, we can find there exsits block, ringing and Posterizing artifacts. The reason for these is using DCT ,sharp edge , shifting form rgb to YcbCr and quantization during the Mpeg.

And for Foreman the effect are more obvious than the Akiyo , and the stationary is more clear than the moving area. This is the error using motion prediction whose inaccuracy leads to the unclear and incontinuity.

(3) Comparision from frame to frame

We can find the bitrates and PSNR varies from frame to frame. This is because there are no rate control that the algorithm at first use smaller qscale which stands for high quality, then to fit the bitrates requirements to increase the qscale which leads to the lower PSNR and variable bitrates.

2.4.2 Gop effect

2.4.2.1： inserting B frame

(1) Comparision between using B frame and previous

a: Bitrates:

When we use the B-fram , the lower bitrates compared to the bitrates using no b frame. The reason for that is that the motion vectors for b-frame are predicted from both backward and forward, which are accurate and lead to smaller residues that we can get higher compression efficiency and lower bitrates for same quality;

b: PSNR

For PSNR, the change is smaller and we can find for Foreman ,the PSNR decrease a little and for Akiyo, the PSNR increase a little. The reason is that firstly, we using the highly accurate motion vector to get lower bitrates , so the PSNR should not be affected much. Secondly, the b-frame is more accurate, thus for foreman, the accurate meaning more information of high frequency is got and removed while for akiyo. The more information of low frequency is reserved.

c: Coding time

For time using, when we insrease the b-fram used , the coding time also increased. This is because the b-frame is based on both back and forward prediction which require more calculation and

leads to the longer coding time.

(2) performance differences Comparision between different b-frame used of two stream

In general , when the b-frame increase , the more accurate the motion vector would be and lower bitrates would be got. For akiyo , it is true and we get lowest bitrate when using 3 b-frame. However, for foreman, when using 3 b-frame, the bitrates increase.

The reason for that is the Akiyo is relatively stationary and the motion vectors are small while the motion vectors in foreman are big , and when we got accurate motion vector we sacrifice the bitrates to save motion vectors. When the effects of motion vector is larger than the saving process , the bitrates increase.

The PSNR also change accordingly , but the change is smaller and the quality is relatively constant.

So , the number of b-frame would affect the performance of the compression and the effect is related to the video source. When compressing relatively stationary video ,we can choose larger b-frame number ,vice verse , choose lower number.

2.4.2.2: Comparision between Different Gop size

(1)Performance differences between Different Gop size

In general, when the Gop size increase, we got lower bitrates and higher PSNR.

We use smaller I-frame number. Because the bits in I frame is larger than the B-Frame ,so when we use smaller I-frame number, we get low bitrates for same quality.

For PSNR, considering the value in P-frame is the residue and relatively small. So the effect of DCT and quantization on the P-frame is smaller than the I -frame which meaning the use of smaller numberof I -frame leads to the increase of PSNR.

For time using, when we insrease the Gop size used , the coding time increase. This is because the we use less I frame ,and less p-frame , thus do leass motion estimation and less calculation.

(2)performance differences between two stream

We find the decrease of the Akiyo is much quicker than Foreman when we use larger gop size.

This is because Akiyo is relatively stationary and different frames are highly related ,so the residue in a large gop can still be small.But for Foreman, the image move fast and the connection between frame and frame is low. Thus using large GOP size may leads to large residue in the P-frame which may compromise the effect of the saving from using less I frame.

(3) How to choose the GOP size

In general, when we use increase the gop size ,we get lower bitrates. But it is not sure and is related to the information of the video stream.

For video with slow motion activity, the different frames are highly related thus the large gop size still preserve the small residue in p-frame ,meaning we still cam make accurate motion prediction in large gop size. At this moment,we can choose higher gop size to decrease the bitrates;

For video with fast motion activity, the different frames are less related thus the large gop size may leads to large residue in the P-frame which may compromise the effect of the saving from using less I frame, meaning we cam not make accurate motion prediction in large gop size. At this moment,we can choose relatively smaller gop size to keep low of the bitrates;

Besides, we use gop because it avoid the error preservation from previous I-frame and achieve the random access ,so the gop size still should not be too large.

2.4.2.3: Comparision between Different Bitrates

(1) Performance between between Different Bitrates

when Bitrates increase,we get higher bitrate ,higher PSNR and larger coding time.

This is because we use more bits on frame in average ,which provides high quality. We need to make more accurate motion estimation and larger calculation which leads to the increase of the coding time.

(2): Comparision between different b-frame used

We find in higher bitrates, when we use large number of b-frame, the bitrates decrease, But in lower bitrate, when we use large number of b-frame, the bitrates even increase ,especially for the Foreman stream.

This is because , in lower bitrates, the bits for I frame is also small , as the difference between I , P and B frame is small. So ,the decrease of B-fram from P frame is small ,while we need more code to store the motion vector ,especially for stream with fast motion activity, which result in the increase of the Bitrates.

So, in conclusion ,the choose of the B-frame structure should also consider the bitrate and the video stream motion activity.

4: Comparision between subjective feeling and objective PSNR

The subjective feeling of the video is not always with PSNR . For example, the larger PSNR using More B-Frame give the bad watch , and using larger gop size leads to discontinuity playing while increase the PSNR. In bitrate change , the lower PSNR can give almost same view feeling.

The reason for above: B-frame may preserve the original block effect and enhance it, which leads to the bad watch feeling ; Large gop size distort the relation between the frame and frame by allocation them into gop when it does not compatible with the original; The lower bitrate is achieved by remove the information in high frequency while human is not sensitive to.

In conclusion , the subjective feeling is related to many reasons including the illuminate, color composition and motion speed , human is less sensitive to high spatial frequency and temporal frequency information. while the PSNR only represent the difference of the pixel value and it can not represent the subjective feeling.

2.4.3 Motion Estimation

2.4.3.1:Comparision between full-pixel and sub-pixel

The sub-pixel motion estimation has less (0,0) than full-Pixel and are based on the sub-pixel .

It is more accurate than full-pixel ,considering the minimum SAD could not be the full-pixel and the sub-pixel provides the more accurate motion prediction.

2.4.3.2:Comparison between Akiyo and Foreman

The motion vectors of Akiyo is much smaller than the motion vectors of foreman, and the motion vectors (not equal to (0,0)) of Akiyo are more uniformly distributed in different direction while the motion vectors (not equal to (0,0)) of Foreman are maily distributed horizontally and in right direction.

From the video we can the Akiyo is relatively stationary which leads to the small motion vector and almost is concentrated at (0,0) ;

While the Foreman has fast motion activity and mainly move to the right. Thus it has bigger motion vector than Akiyo.

2.4.3.3: Comparision between Different Bitrate size

For foreman, We can find when bitrates decrease from 300k , at first , there are less value in (0,0) and there are larger motion vector in average. And then , there more value in (0,0) and there are smaller motion vector in average.

This is because , at first ,when we try to low down the bitrates , we need more accurate motion vectors (rather than zero motion vectors) to get less residue; but when we continue to find lower bitrates, the motion vectors take more portion. Thus , we need to low down the size of motion vector and leads to more value in (0,0).

The motion vectors of Akiyo change slowly than Foreman for Akiyo is relatively stationary thus in high bitrate the most motion vectors are still be (0,0);

2.4.3.4:Conclusion

a: The motion vector are mainly centered at the (0,0) , which meaning the (0,0) have strong possibility to be the predicted motion vector.

b: The motion vectors is highly related to the video stream , for low motion activity , the motion vecotrs is small and for fast motion activity , the motion vecotrs is large.

c: The size of motion vector codes change along with the bitrates (mor zero vectors) , at first ,when we try to low down the bitrates , we need more accurate motion vectors (rather than zero motion vectors) to get less residue; but when we continue to find lower bitrates, the motion **vectors take more** portion. Thus , we need to low down the size of motion vector and leads to more value in (0,0).

Problem 3: Rate control

3.1 Abstract and Motivation

Considering we use the I-frame, P-frame and B-frame, while there are more bits in I-frame than P-frame and more bits in P-frame than B frames. So the out biterate can not be a constant which leads to the problems in the reciever. We prefer a constant or appoxiamate constant byterate for the transfer.Thus ,we need to do the rate control after the encoder.

One common method of rate control is to use a buffer. The codes are sent into a buffer at first ,and then be sent from the buffer in a constant bitrate.

There are also problems with the buffer, when input speed is high and buffer size is not enough,there would occus the overflow problem. When output speed of buffer is much larger than the input,there would occus the problem of underflow.

3.2 Aproach and Procedures

A. Buffer value

using the command

ffmpeg -f rawvideo -vcodec rawvideo -s 352x288 -r 25 -threads 2 -pix_fmt yuv420p -i foreman_cif.yuv -c:v mpeg2video -me_method epzs -r 25 -g 10 -b:v 250k -maxrate 250k -minrate 250k -bufsize 128k -psnr -vstats_file foreman_state_b_ff.txt foreman_buff.mp4

-bufsize 128k: set the buffer size qual to 128kbits.

B. Rate control method

1. Summarize the rate control scheme

The key point of the rate control is to control the input frame bits by changing the QP value. In this paper, it dynamically adjusts the quantization parameter for each frame by considering coded frame quality, allocated transmission bandwidth, buffer occupancy and scene changes. In the proposed strategy, the QP is divided into the basic quantization scale (BQS), which is used to maintain the constant level of visual quality, and the adjustable quantization scale (AQS), which is used to fit the transmission rate and avoid overflow and underflow conditions.

The main flow of the rate control is shown below:

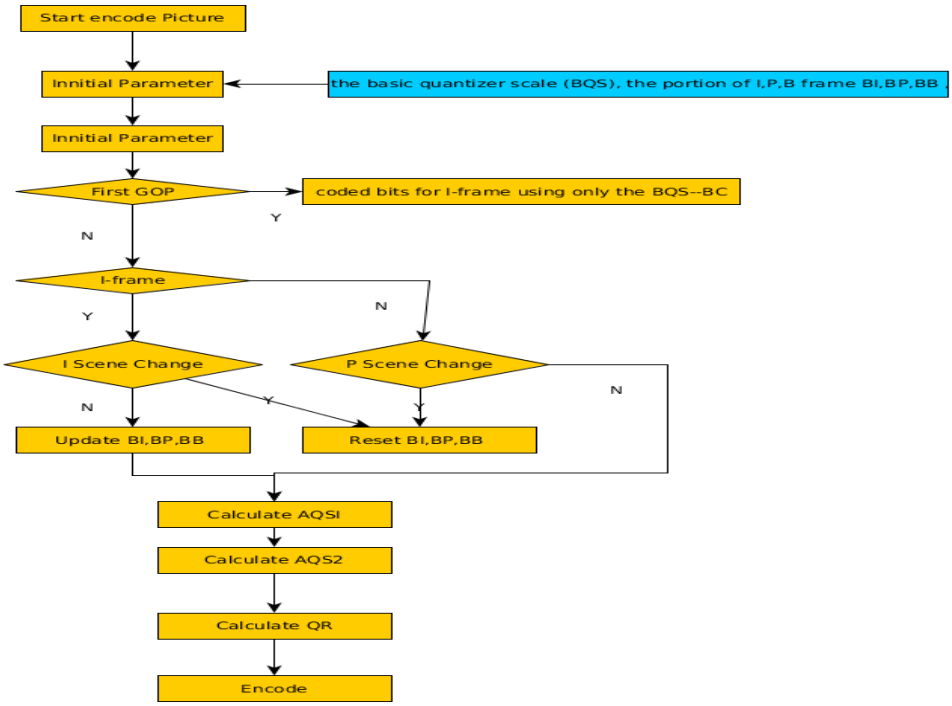


Figure 9 main flow of rate buffer conrol

calculate the AQS_I : each coded bits for I-frame using only the BQS could be get using

$BC'_x = BC'_I * B_x/B_I$, then use

$$Prate = \frac{BC'_I + BC'_P \times N_P + BC'_B \times (N_{GOP} - 1 - N_P)}{N_{GOP}} \times fram_rate \quad T_X = BC'_X \times \frac{T_{rate}}{P_{rate}} \quad AQS1_x = \frac{P_{rate} - T_{rate}}{buffersize} \times S_x$$

AQS1 is tuned per GOP.

calculate the AQS_2,

$$AQS2_x = \frac{outcome - target}{buffersize} \times S_x \quad outcome = \sum_{i=1}^{n}(S_i - T^i_{rate}/framerate) \quad outcome = \sum_{i=1}^{n}(T_i - T^i_{rate}/framerate)$$

I scene chage judegement B_I,B_P,B_B updated B_I,B_P,B_B reset

$$\left|\frac{BitCount_I - T_I}{buffersize}\right| > \delta \qquad \begin{cases} B_I = U_I/U_B \\ B_P = U_P/U_B \\ B_B = 1 \end{cases} \qquad \begin{cases} B_I = B^n_I \times BC'_I/ICount \\ B_P = B^n_P \times BC'_I/ICount \\ B_B = B^n_B \end{cases}$$

2. Coding implemention

Insert the related code at the mpegvideoenc.c file.

(1) the main process

```
BCI=ROP;//Recalculate the BCI
BCP=ROP*BP/BI;//Recalculate the BCP
Prate=(double)(BCI+BCP*9)*25/10;//Recalculate the Prate
 printf("\n Prate %f ROP %ld b->qscale %d \n",Prate,ROP,b_qscale);
 AQS1= (Prate-Trate)/128000; ////Recalculate the AQS1
 printf(" the bits of the AQS1 = %f\n ", AQS1);
 if (s->pict_type==AV_PICTURE_TYPE_I) T_d=(BCI*Trate)/Prate;  //the desire T-bits
 else T_d=(BCP*Trate)/Prate;
 out_bits+=(double)(s->frame_bits-Trate/25);
 tar_bits+=(double)(T_d-Trate/25);
```

(2) update process

```
   BI=(I_total*P_count)/(P_total* I_count);
```

(3)Scen change judement

```
   if( (( (((ROP*Trate)/Prate)-put_bits_count(&s->pb))>128000*0.25 || (((ROP*Trate)/Prate)-put_bits_count(&s->pb))<(-
128000*0.25) )) &&P_count!=0 )  //scen change
  {
   printf("\n a---------------------------------- \n");
   BI=2;  //for Akiyo BI default = 5 ; for foreman BI default = 2
   BCI=ROP;
   BCP= ROP*BP/BI;

}
```

3.3 Results

A. Buffer value

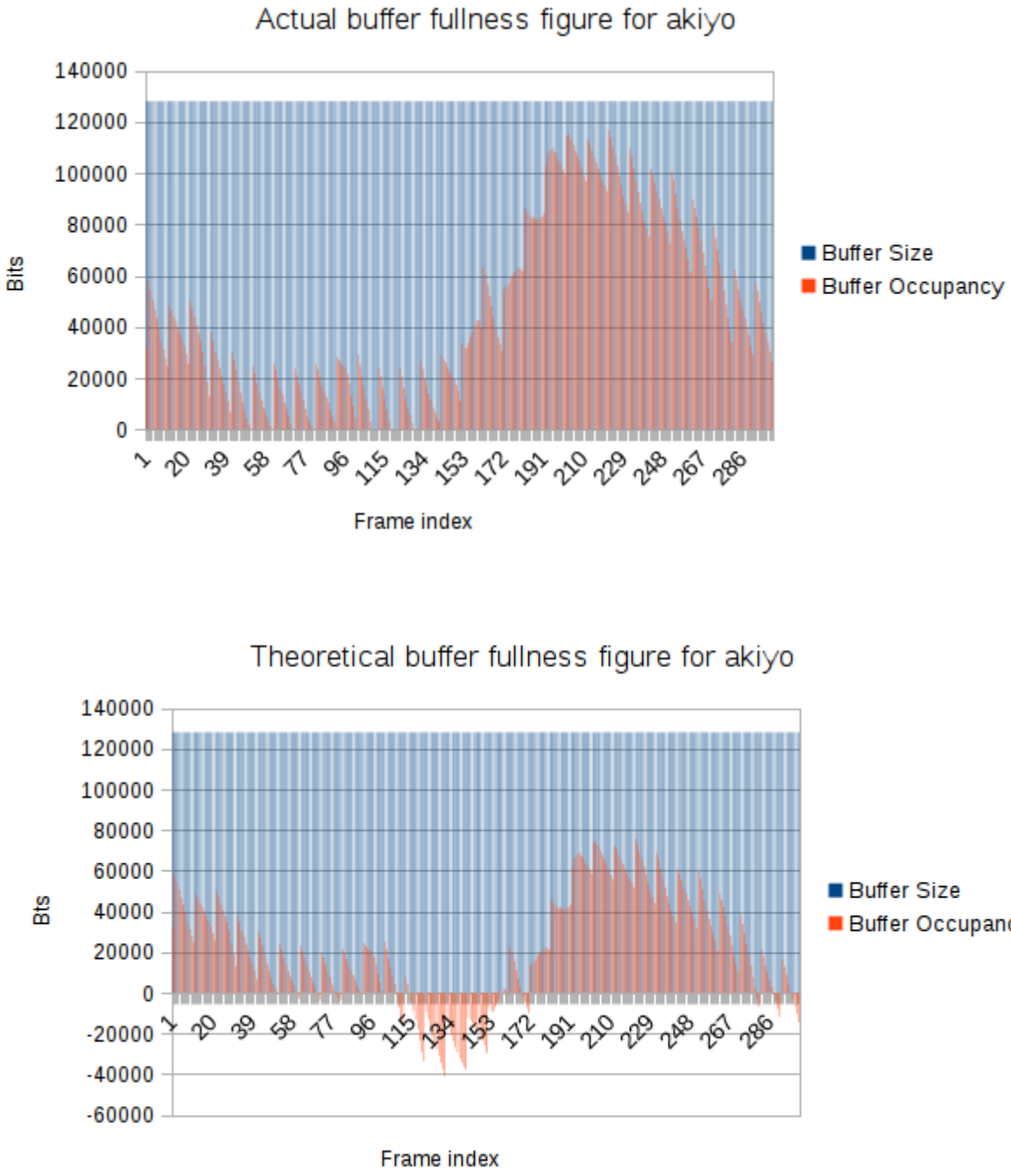


Figure 10 Buffer fullness for Akiyo

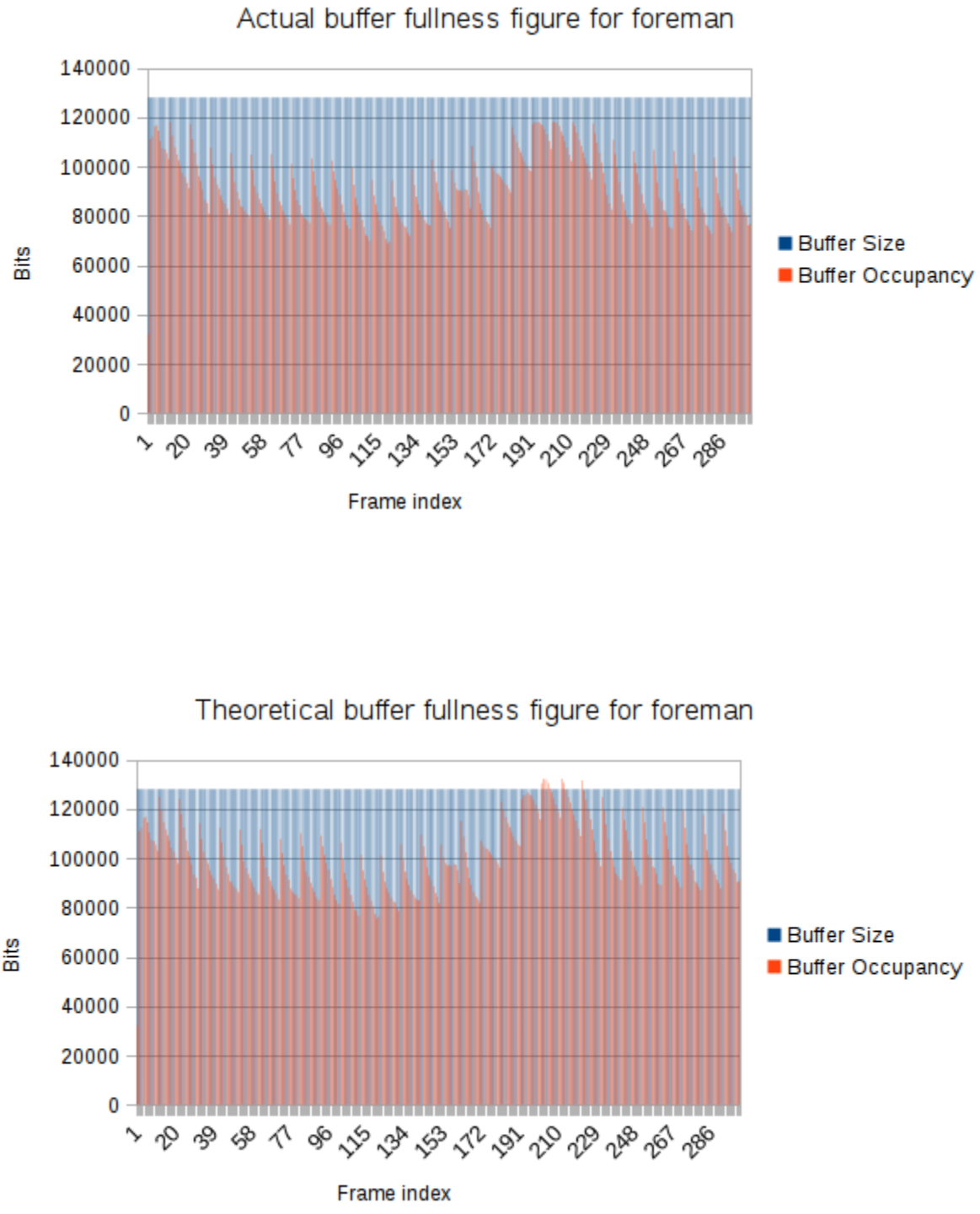


Figure 11 Buffer fullness for Foreman

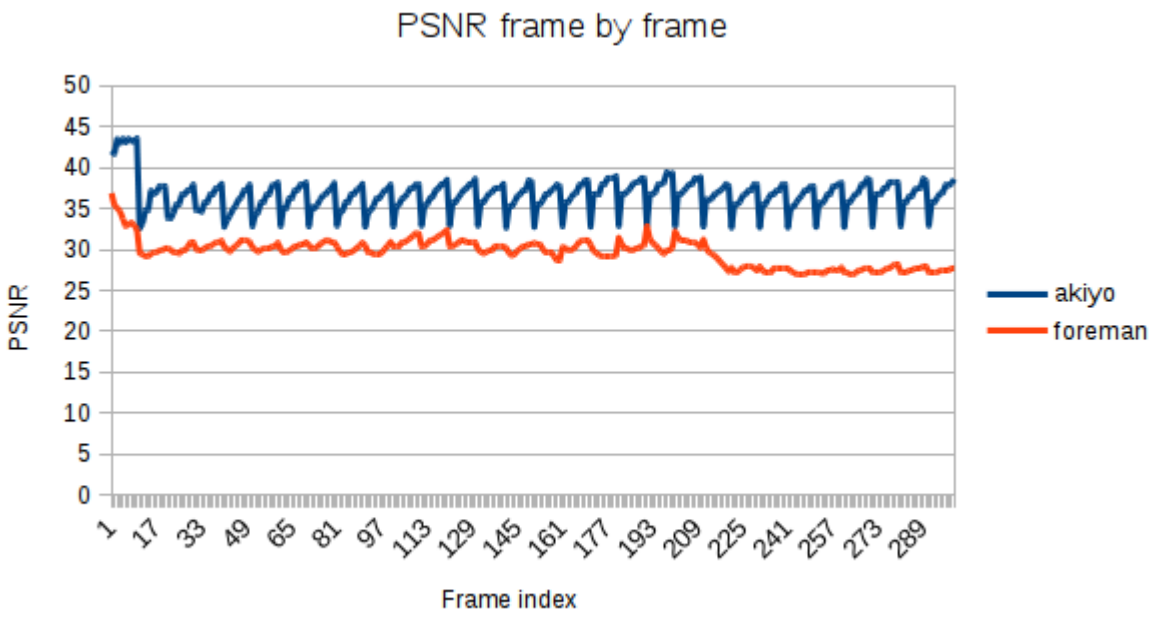


Figure 12 PSNR for two stream Frame by Frame

B. Rate control method buffer fullness and PSNR

1:Akiyo

comparison with using and not using rate control

Bitrate = 250k

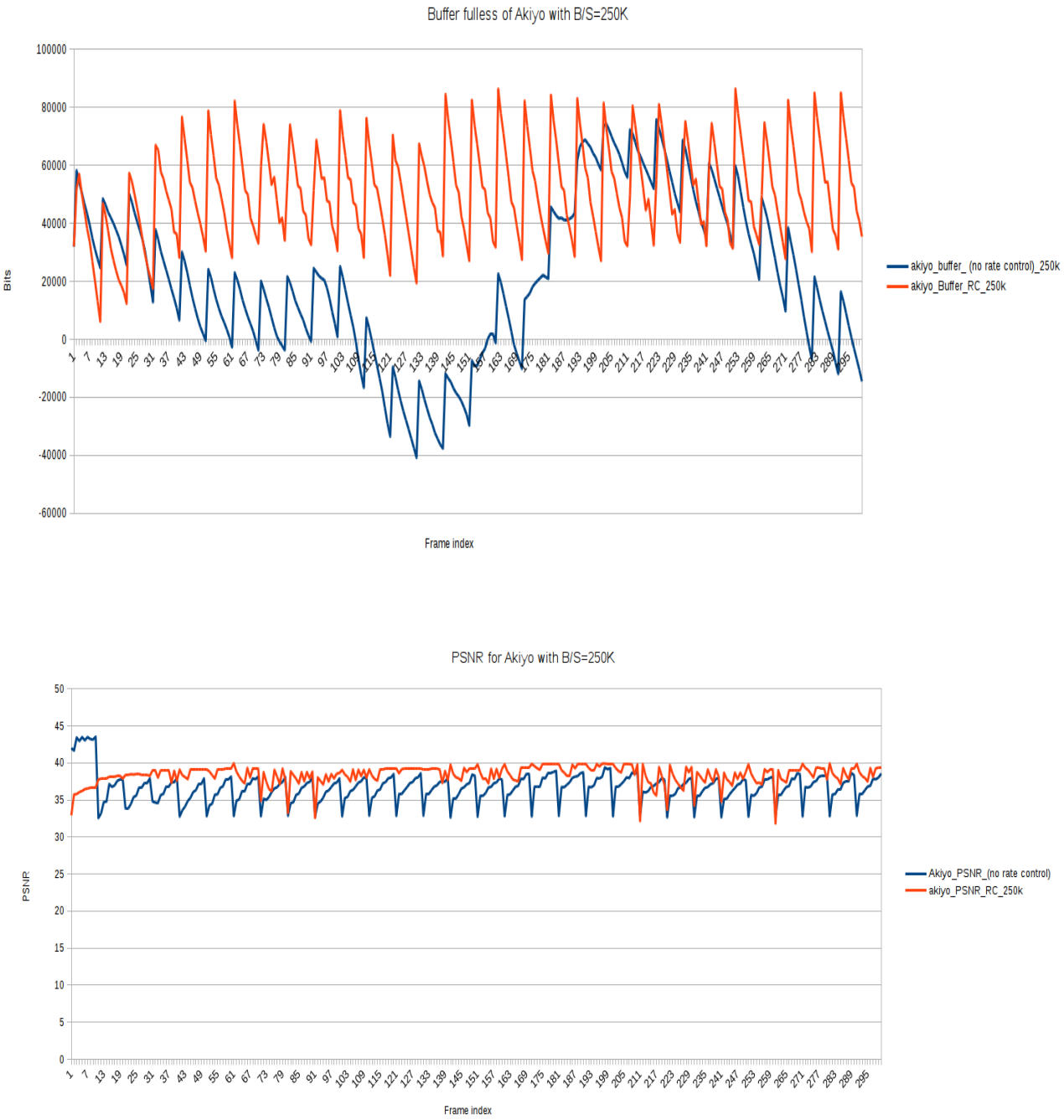


Figure 13 Rate Control Comparison for Akiyo

2:Foreman

comparison with using and not using rate control :

Bitrate = 250k

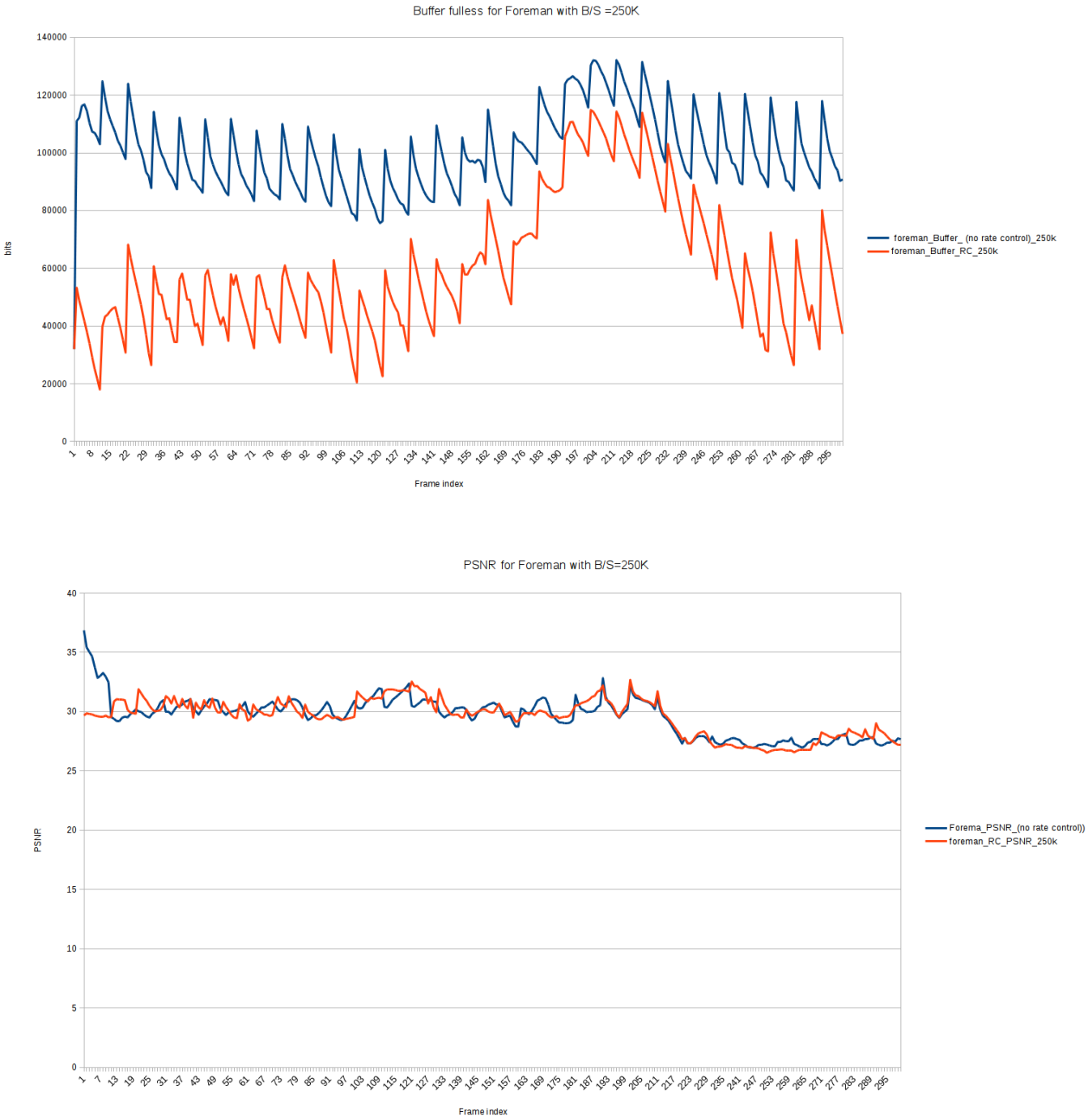


Figure 14 Rate Control Comparison for Foreman

B. Robustness test

1:Akiyo

Comparison with different bitrates : Bitrates from 100k to 400 k

Overflow occurs at Bitrates around 100k

Underflow occurs at Bitrates around 400k

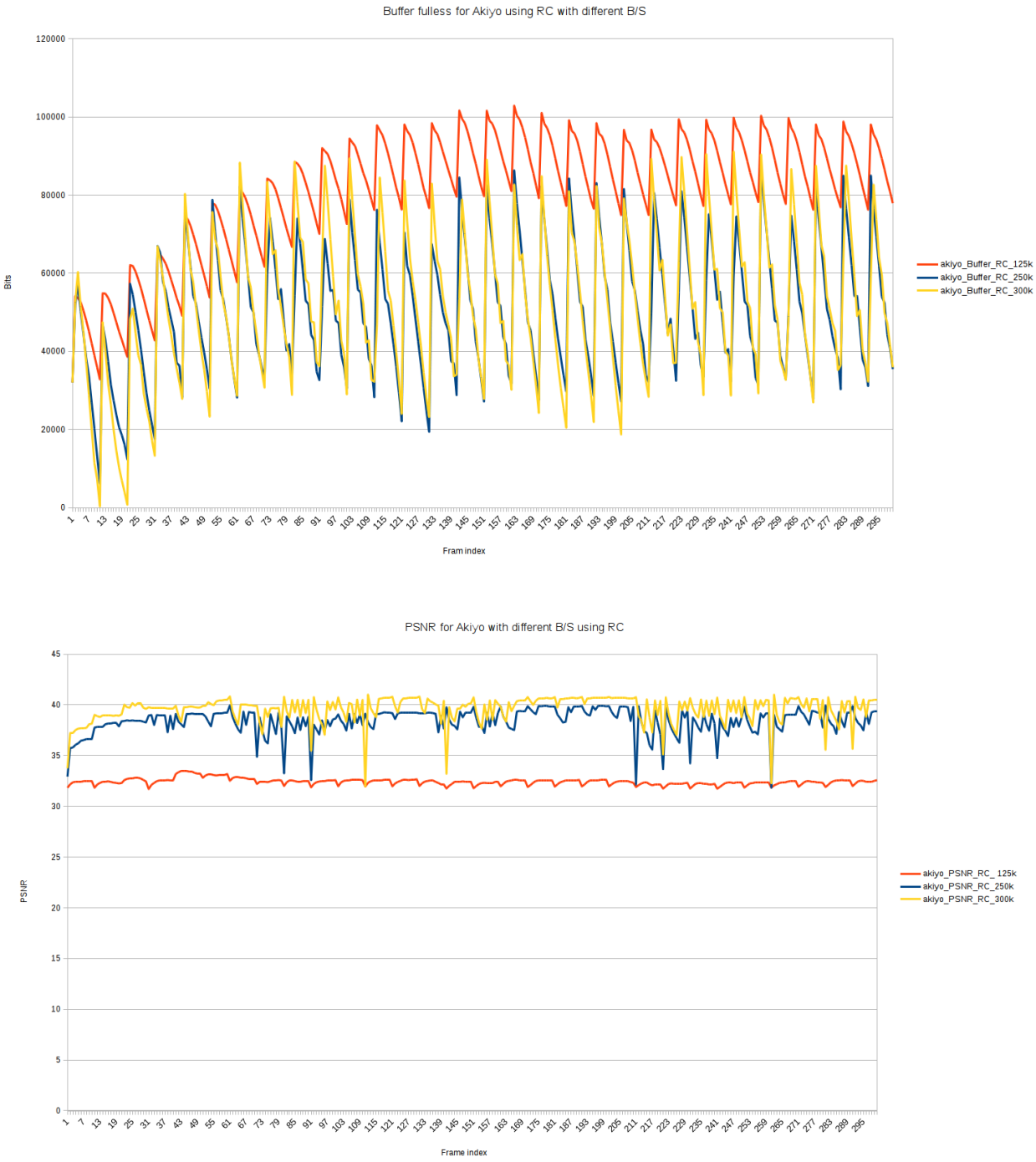


Figure 15 Rate Control Robustness test for Akiyo

2:Foreman

Comparison with different bitrates : Bitrates from 230k to 450 k

Overflow occurs at Bitrates around 230k

Underflow occurs at Bitrates around 450k

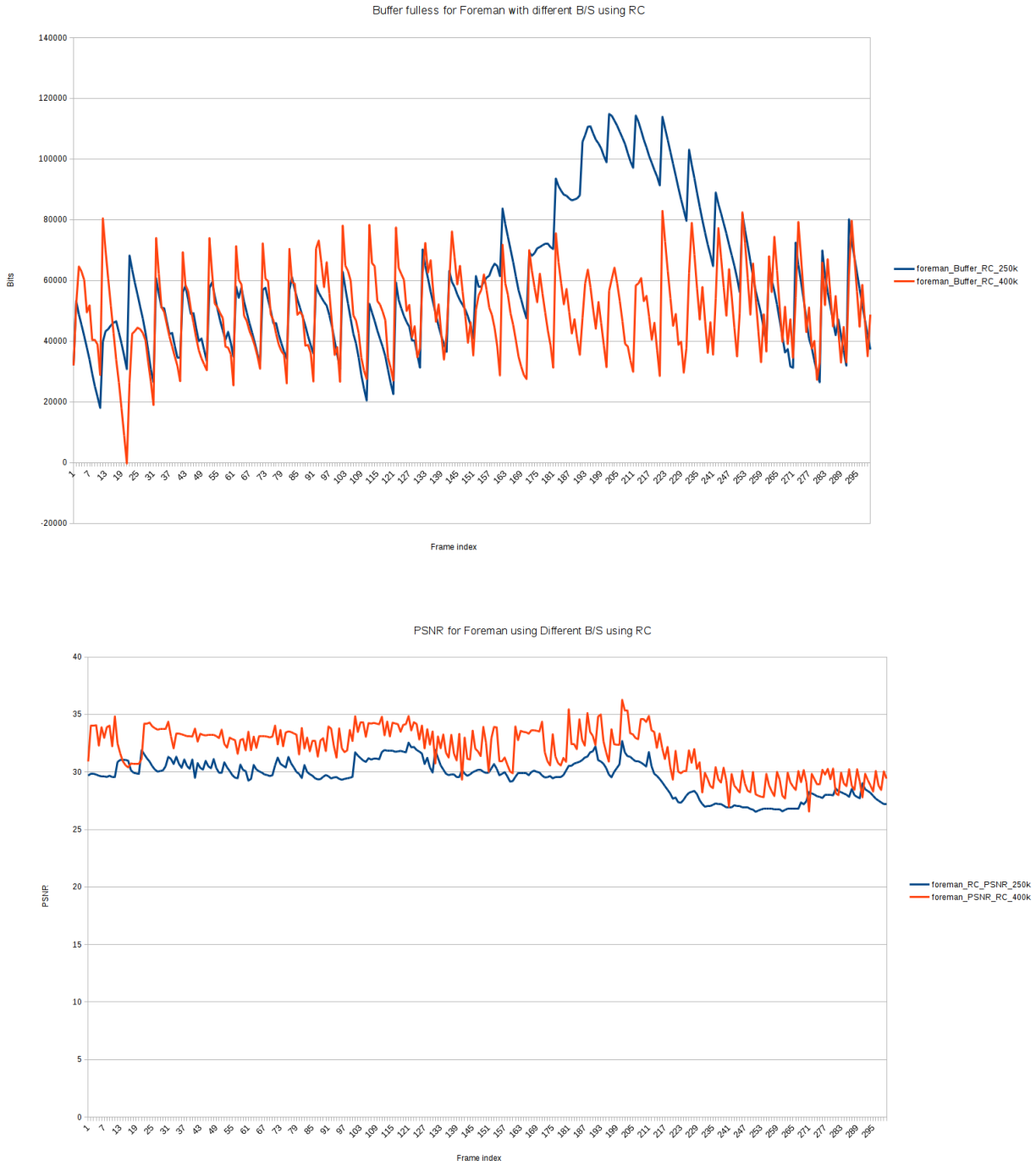


Figure 16 Rate Control Robustness test for Foreman

3.4 Discussion

3.4.1 Buffer fullness and PSNR without rate control

For both , the Buffer occupancy increase large number when receive I-frame and decrease with the receive of the P-frame.

This is because the I-frame is coded without motion prediction while the P-frame is the value residue, thus, the bits in the I-frame is larger than the P-frame;

For Akiyo, we can find there occurs underflow situation.

This is because the Akiyo has slow motion activity, thus the P-frame has small residue and the motion vectors are small, So the input datas are also small. So the underflow occurs for the input speed is smaller than output causing the data in buffer become zero.

For Foreman, we can find there occurs overflow situation.

This is because the Foreman has quik motion activity, thus the P-frame has relatively larger residue and the motion vectors are large, So the input datas are also large. So the overflow occurs for the input speed is bigger than output causing the data in buffer become full.

3.4.2 Comparison of Buffer fullness and PSNR using rate control

From the figure ,we can see the Rate control avoid the overflow and underflow situation while no decreasing the PSNR.

For Akiyo, the buffer occupancy is stable around 64k and keep the PSNR almost constant.

For Foreman, the buffer performance avoid the overflow. There are a rapid increase of the buffer occupancy around the 200 frame ,it is the place scene change ,considering I -frame scene change there are not much increase in the buffer occupancy. The PSNR is still kept almost constant.

3.4.3 Robustness test

for the Akiyo, Overflow occurs at Bitrates around 100k , Underflow occurs at Bitrates around 400k;

for Foreman, Overflow occurs at Bitrates around 230k,Underflow occurs at Bitrates around 450k;

Thus ,we can see the higher bitrates ,the underflow is more likely to occur , while the lower bitrates , the overflow is more likely to occur.

This because the bitrates meaning the outpeed from the buffer, the higher the bitrate the high speed of output, the buffer is less likely to be full while increase the risk of become zero occupancy.

3.4.4 Conclusion

Rate control in important in the mpeg coding for the different bits in different types of frame, the key of rate control is to control the input frame bits which should be linked to the buff size and bitratesthe input video stream.