简单错误对 H.264 视频编码性能影响的研究

张 昊[1]，占海燕[1]，邓月堂[2]
（1. 中南大学 信息科学与工程学院，长沙 410075；2. 腾讯，广东 深圳 518057）

摘 要：研究了简单错误对 H.264 视频编码器的影响，目标是为编码器测试的工作提供依据。通过在编码器中采用变异算子引入简单错误，可以分析简单错误对 PSNR 和主观质量的影响。实验结果显示，对大部分植入的简单错误没有明显地降低编码器的 PSNR 和主观质量。因此编码器的测试需要研究人员更加深入的探索。

关键词：视频编码器；软件测试；简单错误；变异算子；编码性能
中图分类号：TP391 **文献标志码**：A **文章编号**：1001-3695(2012)10-3948-04
doi:10.3969/j.issn.1001-3695.2012.10.093

Research on impact of simple faults on H.264 encoder performance

ZHANG Hao[1], ZHAN Hai-yan[1], DENG Yue-tang[2]
(1. *School of Information Science & Engineering, Central South University, Changsha* 410075, *China*; 2. *Tencent Inc., Shenzhen Guangdong* 518057, *China*)

Abstract: To facilitate encoder testing, this paper studied the impact of simple faults on H.264 video encoders. By introducing simple faults generated by mutation operators into the encoder, it analyzed the impact of mutation faults on video PSNR and perceptual quality. Experiment results show that most simple faults do not lead to considerable degradation on PSNR and perceptual quality. Hence encoder testing requires the deeper investigation.

Key words: video encoder; software testing; simple fault; mutation operator; encoder performance

0 引言

视频编/解码技术越来越发达，目前已得到广泛应用的 H.264国际标准，其压缩效率相对于以前的标准提高了很多，而相应地复杂度也越来越高。随着各种硬件平台和相应 SIMD 指令的广泛应用，厂家往往需要支持视频编码器的不同优化版本，这使得其维护和测试也更加困难。

对工业界来说，视频编/解码器的测试是一个影响产品质量的重要问题，但学术界对其研究还不是很多。虽然软件测试技术已经广泛应用，但是如何将传统的测试技术有效地应用于视频编/解码器的测试，以及视频编/解码器的测试是否存在特殊性和需要开发特殊的测试技术，这方面的研究成果尚不多见，目前的研究主要集中于测试解码器的兼容性或优化模块的单元测试[1]。

视频压缩分为编码和解码。对于正确编码后产生的码流，所有厂家的解码结果应该都是一样的。因此，对于需要测试的解码器而言，如果解码结果跟标准软件 JM 不一致，就可以认为软件中存在错误。根据这个特点可以设计不同的测试方法，在不分析源代码的情况下，都可以进行兼容测试。对编码器而言，标准并未规定唯一的编码算法，因此不同厂家的编码器算法和产生的码流允许不同，这样就很难确定编码器是否存在错误。以上分析说明，相比解码器，由于缺乏统一的标准输出，编码器的测试技术难度更大，这也是本文将要研究的内容。

虽然编码器的错误可能很难精确地测试和定位，但相对不包含错误的编码器，其也将引起客观和主观质量的变化。一个很自然的想法就是：假设待测试的编码器采用了跟标准软件 JM 相同的算法，它们的编码效率应该是近似的，那么有没有可能通过观察客观和主观视觉质量的变化而确定待测编码器有无错误呢？本文将通过一些实验来初步探讨这个问题。

编码器的错误可能是算法设计层次的错误，也可能是程序员在编写过程中的代码出错。本文只研究后一种错误。由于研究人员往往很难收集到编码器中的真实错误[2]，本文将在编码器中采用变异算子植入自动生成的错误。采用变异算子植入错误的优点是可以由软件系统地、大量地自动植入，具有可重复性，便于进行统计分析[3]。另外，本文将采用测试中常见的两个原则，即程序员的能力假设（competent programmer hypothesis）和耦合效应（coupling effect）：第一个原则假定程序员写的代码接近正确，即使出错也只会产生比较简单的错误；耦合效应即表示如果某些测试用例能够发现简单错误，那么它们也能够在很大程度上发现复杂错误[4,5]。基于这两个原则，在本文的实验中只考虑由变异算子产生的简单错误，然后测试错误对编码性能的影响，由此判断根据性能下降指标推测编码器是否包含错误的可行性。

1 错误的生成与植入

为了让植入的错误更具有普遍性，在实验中采用了变异测试工具来自动生成错误[6]。本文采用文献[7]中列出的变异算子类型（表1），然后将各种类型的错误植入到 JM 编码器中，

收稿日期：2012-03-05；**修回日期**：2012-04-09

作者简介：张昊(1978-)，男，湖南长沙人，副教授，博士，主要研究方向为多媒体信号处理与通信、软件测试技术及其应用、科技评价理论(hao@csu.edu.cn)；占海燕，女，硕士研究生，主要研究方向为视频压缩中的软件测试技术；邓月堂，男，博士，主要研究方向为软件测试技术.

对含有错误的编码器编码后的 PSNR 与无错的编码器进行比较,并观察该错误对整个编码过程的影响。

表 1　变异错误的类型和介绍[7]

类型	操作名	介绍
算术运算	AORB	算术运算符替换(二元)
	AORU	算术运算符替换(一元)
	AORS	算术运算符替换(自增或自减)
	AOIU	插入算术运算符(一元)
	AOIS	插入算术运算符(自运算)
	AODU	删除算术运算符(一元)
	AODS	删除算术运算符(自运算)
关系运算	ROR	关系运算符替换
条件运算	COR	条件运算符替换
	COI	插入条件运算符
	COD	删除条件运算符
移位运算	SOR	移位运算符替换
逻辑运算	LOR	逻辑运算符替换
	LOI	插入逻辑运算符
	LOD	删除逻辑运算符

本文主要对 JM 中的帧内预测、去方块滤波、运动估计(本文采用 JM 中的六边形快速搜索)、亮度插值四个部分进行了实验,在这四个部分中植入错误生成标准编码器的变异体。实验的过程如下:

a)用变异测试工具对帧内预测函数进行处理,生成表 1 中的各个类型的所有可能出现的错误。

b)生成的错误数量很大。为了简化测试过程,从所有的错误中随机选取 20 个,在标准编码器对应地逐个植入这些错误,生成包含单个错误的编码器。

c)用错误的编码器对多个视频序列进行编码,观察主观和客观质量,与正确的编码器相对比。

2　实验结果

2.1　帧内预测

为了更好地说明帧内预测中的简单错误对编码器的影响,对实验序列设置为全 I 帧编码。其他编码参数如下:编码帧数 100,帧率 30 Hz, QP = 28。实验对三个 CIF 序列:foreman、mother&daughter、container 进行编码。帧内预测部分选择的 20 个错误及其编码 PSNR 变化情况对实验数据进行分析,如表 2 和图 1 所示。当帧内预测部分中包含简单错误时,编码结果大约 70% 的 PSNR 变化大于 0.5 dB。

2.2　去方块滤波、运动估计、亮度插值

去方块滤波、运动搜索和插值都是编码的重要流程之一,此部分的实验目标在于探索在这三个部分中包含有简单错误时编码器的性能。运动估计采用六边形搜索算法。对每个部分随机植入 20 个错误,用变异编码器对系列进行编码。实验的主要参数如下:编码层次为 baseline,编码帧数 100,帧率 30 Hz,QP = 28,搜索范围是 5。实验序列不变。这三个部分的结果数据分别在表 3 ~ 5 和图 2 ~ 4 中,可知在这三个函数中,80% 以上单个的简单错误对整体 PSNR 的影响在 0.5 dB 范围内。因此,除了帧内预测模块,变异算子产生的简单错误对 PSNR 产生下降的幅度不是很大,那么采用观察客观质量下降的方法不一定能够有效检测编码器的错误。

表 2　帧内预测模块中引入的错误和引起编码 PSNR 的变化情况

序号	类型	具体错误		PSNR 情况(-表示下降)/dB foreman/container/mother
		正确代码	错误代码	
1	AORB	currMB→pix_x + block_x	currMB→pix_x * block_x	-26.159/-23.170/-18.809
2	AORB	best_ipmode - 1	best_ipmode * 1	-24.263/-22.086/-33.679
3	AORB	best_ipmode - 1	best_ipmode + 1	-25.886/-26.188/-32.182
4	AOIU	currMB→pix_x	-currMB→pix_x	-26.693/-24.916/-32.274
5	AOIU	* cost += cost4x4	* cost += -cost4x4	-26.022/-23.808/-23.245
6	AOIU	non_zero[0]⊨currSlice→ mode_decision_for_I4x4_blocks	non_zero[0]⊨-currSlice→ mode_decision_for_I4x4_blocks	0/0/0
7	AOIS	b4	++b4	-26.436/-24.112/-33.631
8	AOIS	b8	++b8	-26.749/-23.773/-33.752
9	AOIS	cost8x8	--cost8x8	-29.181/-27.359/-34.196
10	AOIS	pic_pix_y	--pic_pix_y	0/0/0
11	AODU	(char) - 1	(char) 1	-25.891/-22.239/-20.583
12	AODU	best_mode = -1	best_mode = 1	-16.007/-17.913/-16.325
13	COD	! currSlice→sp2_frame_indicator	currSlice→sp2_frame_indicator	-0.289/-0.335/-0.3434
14	COR	ipmode == HOR_PRED ‖ ipmode == HOR_UP_PRED	ipmode == HOR_PRED^ipmode == HOR_UP_PRED	0/0/0
15	COR	upMode < 0 ‖ leftMode < 0	upMode < 0&&leftMode < 0	0/0/0
16	COI	(slice_type == SP_slice)	(slice_type == SP_slice)	0/0/0
17	COI	(ipmode == DC_PRED)	! (ipmode == DC_PRED)	-8.369/-2.622/-1.8635
18	LOI	b8&0x01	~b8&0x01	-25.944/-23.401/-33.192
19	LOI	top_block. avaible	~top_block. avaible	-23.033/-28.951/-27.598
20	SOR	1 << b8	1 >> b8	-22.816/-28.849/27.565

3　简单错误产生的主观质量影响

对变异编码器的输出比特流进行解码,解码出来的视频序列只有很少的一部分引起了视觉上的差异。比如,图 5 给出了表 1 中编号 3 错误产生的重构图像和不包含错误的重构图像的对比。

图 5(a)(b)表明,含有错误的编码器对其主观质量没有明显的变化,其 PSNR 下降也很小;然而由(c)中显示的差值图像中可见两个图像中的像素还是存在很大的区别,只是这些差别在重构图像中不易发现。

同样在表 1 中,包含编号 7 错误的编码器和不含错误编码器的重构图像的对比如图 6 所示。在这一组图像中,由错误的编码器得到的图像有明显的错误,可以在视觉上清楚地发现这些错误。

表 3 去块滤波器中引入的错误和引起编码 PSNR 的变化情况

序号	类型	具体错误		PSNR 情况(-表示下降)/dB
		正确代码	错误代码	foreman/container/mother
1	AOIS	MbQ→DFDisableIdc == 1	++ MbQ > DFDisableIdc == 1	-0.027/0.0055/-0.025
2	AOIS	MbQ→DFDisableIdc == 1	-- MbQ→DFDisableIdc == 1	0/0/0
3	AOIS	edge < 4	++ edge < 4	0/0/0
4	AOIS	MbQ→cbp == 0	MbQ→cbp += = 0	0/0/0
5	AOIS	filterNon8x8LumaEdgesFlag[edge]	filterNon8x8LumaEdgesFlag[edge ++]	0/0/0
6	AOIU	! (MbQ→luma_transform_size_8x8_flag)	-! (MbQ→luma_transform_size_8x8_flag)	0/0/0
7	ROR	p_Vid→structure! = FRAME	p_Vid→structure == FRAME	0.0005/0/3E-05
8	ROR	p_Vid→structure! = FRAME	p_Vid→structure = FRAME	-0.01/0.0076/-7E-04
9	ROR	MbQ→DFDisableIdc == 1	MbQ→DFDisableIdc ! = 1	0/
10	ROR	edge > 0	edge > = 0	-0.024/0.0035/-0.001
11	ROR	edge > 0	edge == 0	0/0/0
12	ROR	edge_cr > = 0	edge_cr > 0	-9E-05/0/0.0049
13	ROR	edge_cr > = 0	edge_cr ! = 0	-0.027/0.0055/-0.025
14	COR	p_Vid→mb_aff_frame_flag && MbQ→mb_field	p_Vid→mb_aff_frame_flag ‖ MbQ→mb_field	0.0035/0/0.0011
15	COR	(p_Vid→structure! = FRAME) ‖ (⋯)	(p_Vid→structure! = FRAME) && (⋯)	0.0035/0/0.0011
16	COR	(p_Vid→structure! = FRAME) ‖ (⋯)	(p_Vid→structure! = FRAME)^(⋯)	-0.002/7E-05/0.0059
17	COD	! MbQ→mb_field	MbQ→mb_field	0.0023/7E-05/0.0059
18	COR	(⋯) && MbQ→mb_field	(⋯) ‖ MbQ→mb_field	0/0/0
19	COI	MbQ→mb_field	! MbQ→mb_field	0/0/0
20	LOR	edge&0x01	edge‖0x01	0/0/0

表 4 运动估计模块中引入的错误和引起编码 PSNR 的变化情况

序号	类型	具体错误		PSNR 情况(-表示下降)/dB
		正确代码	错误代码	foreman/container/mother
1	AOIS	pred. mv_x = pic_pix_x + pred_mv→mv_x	pred. mv_x = pic_pix_x + pred_mv→mv_x ++	0.0123/-0.002/-0.007
2	AOIS	search_range	++ search_range	0/0/0
3	AOIU	cand. mv_x = pic_pix_x	cand. mv_x = - pic_pix_x	-0.006/0.0023/0.0052
4	AOIU	p_UMHex - > Bsize[blocktype]	- p_UMHex→Bsize[blocktype]	-0.004/0/0
5	AORB	pic_pix_x + pred_mv→mv_x	pic_pix_x * pred_mv→mv_x	0.0051/0.0002/0.0049
6	AORB	iMinNow. mv_y + p_Vid→spiral_qpel_search[pos]. mv_y	iMinNow. mv_y - p_Vid→spiral_qpel_search[pos]. mv_y	0.0002/0/0
7	ASRS	i += 8	i * = 8	-0.142/-0.103/-0.092
8	ASRS	mcost += (⋯)	mcost -= (⋯)	0/0/0
9	COI	(best. mv_x == iMinNow. mv_x &&best. mv_y == iMinNow. mv_y)	! (best. mv_x == iMinNow. mv_x &&best. mv_y == iMinNow. mv_y)	0/0/0
10	COI	if((ref > 0⋯⋯))	if(! (ref > 0⋯))	-0.028/-0.002/-0.029
11	COR	(center. mv_x! = pic_pix_x ‖ center. mv_y ! = pic_pix_y)	(center. mv_x ! = pic_pix_x&&center. mv_y! = pic_pix_y)	0/0/0
12	COR	ref > 0&&currSlice→structure == FRAME	ref > 0 ‖ currSlice→structure == FRAME	0.0284/0/0.0043
13	LOI	search_range = mv_block→searchRange. max_x > > 2	search_range = ~ mv_block - > searchRange. max_x > > 2	0/0/0
14	LOI	temp_Big_Hexagon_Y[m] += Big_Hexagon_Y[m];	temp_Big_Hexagon_Y[m] += ~ Big_Hexagon_Y[m];	-0.007/0/-2E-04
15	ROR	if(blocktype > 1)	if(blocktype < 1)	0.0187/0/0.0072
16	ROR	if(blocktype > 6)	if(blocktype > = 6)	0.0072/0/0
17	SOR	mv_block→searchRange. max_x > > 2	mv_block→searchRange. max_x < < 2	-8E-04/0/0
18	SOR	i < = (search_range > > 2)	i < = (search_range < < 2)	0.0012/0/0
19	AOIS	center. mv_x = pic_pix_x + mv→mv_x	center. mv_x = pic_pix_x + + + mv→mv_x	0.0152/0/0
20	AOIS	pic_pix_y + (⋯)	(pic_pix_y + + + (⋯)	0/0/0

表 5 亮度插值模块中引入的错误和引起编码 PSNR 的变化情况

序号	类型	具体错误		PSNR 情况(-表示下降)/dB
		正确代码	错误代码	foreman/container/mother
1	AODS	tap0 * (* srcImgA + + + * srcImgD + +)	tap0 * (* srcImgA + * srcImgD + +)	-0.563/-0.352/-0.18
2	AODS	tap1 * (* srcImgB + + + * srcImgE + +)	tap1 * (* srcImgB + * srcImgE + +)	-0.646/-0.76/-0.144
3	AODS	tap0 * (* srcImgA + + + * srcImgD + +)	tap0 * (* srcImgA + * srcImgD + +)	-0.646/-0.76/-0.144
4	AOIS	tap0 * (* srcImgA + * srcImgD)	tap0 * (* srcImgA + * srcImgD + +)	-0.716/-0.307/-0.162
5	AOIS	tap1 * (* srcImgB + * srcImgE)	tap1 * (* srcImgB - - + * srcImgE)	0.0058/0.0014/0.0114
6	AOIS	wBufDst = dstImg[jpad] - IMG_PAD_SIZE_X;	wBufDst = dstImg[jpad + +] - IMG_PAD_SIZE_X;	0/0/0
7	AOIU	tap0 * (* srcImgA + * srcImgD)	tap0 * (- * srcImgA + * srcImgD)	0/0/0
8	AOIU	* iBufDst + + = is;	* iBufDst + + = - is;	0/0/0
9	AORB	* srcImgC + + + * srcImgF + +	* srcImgC + + + * srcImgF + +	-0.217/-0.112/-0.144
10	AORB	tap1 * (⋯)	tap1 + (⋯)	0/0/0
11	AORB	tap1 * (⋯) + tap2 * (⋯)	tap1 * (⋯) - tap2 * (⋯)	0/0/0
12	AORS	(* srcImgA + + + * srcImgD + +)	(* srcImgA - - + * srcImgD + +)	-0.217/-0.115/-0.144
13	AORS	(* srcImgB + + + * srcImgE + +)	(* srcImgB + + + + + * srcImgE)	0/0/0
14	LOI	ypadded_size - img_pad_size_y	~ ypadded_size - img_pad_size_y	0/-0.151/-0.196
15	LOI	* iBufDst + + = is;	* iBufDst + + = ~ is;	-0.647/-0.013/-0.032
16	LOI	* srcImgA + + + * srcImgD + +	* srcImgA + + + ~ * srcImgD + +	-0.087/-0.112/-0.144
17	LOI	tap1 * (* srcImgB + * srcImgE)	tap1 * (~ * srcImgB + * srcImgE)	-0.217/0/0
18	AORB	tap1 * (* srcImgB + + + * srcImgE)	tap1 + (* srcImgB + + + * srcImgE)	0/0/0
19	AORB	tap1 * (⋯)	tap1 - (⋯)	0/0/0
20	AOIS	* iBufDst + + = is;	* iBufDst + + = is + + ;	0/0/0

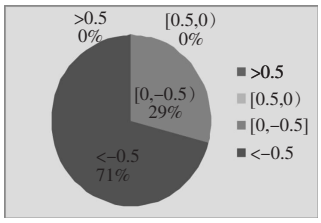


图 1　帧内预测模块
PSNR 变化分布

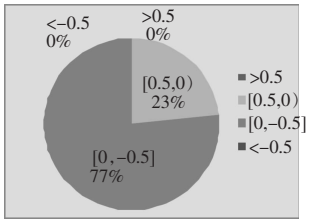


图 2　运动估计模块
PSNR 变化分布

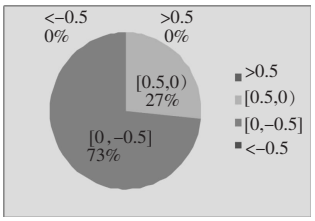


图 3　去方块滤波器
PSNR 变化分布

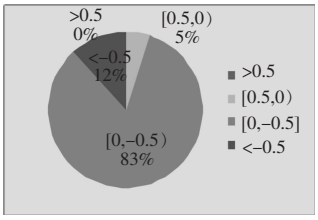
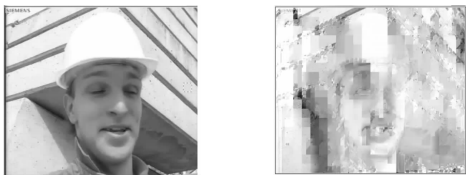


图 4　亮度插值模块
PSNR 变化分布

(a) 无错误比特流解码后的图像　(b) 错误比特流解码后的图像　(c) 对(a)(b)的差值图像进行二值化的结果

图 5



(a) 无错误比特流解码后的图像　(b) 错误比特流解码后的图像

图 6

表 6 中列出了变异错误对主观质量的影响。主观质量由三人观察视频后的平均得分表示。图 7 中显示了主观质量与 PSNR 的变化对比。从图中可以看出,主观视觉质量是与客观质量的变化相吻合的。

表 6　变异错误引起主观质量变化情况

编码视频序列(cif. yvu)	foreman	container	mother-daughter
PNSR 下降的平均值/dB	4.223 4	4.031 1	4.633 2
主观质量(由三人打分平均求得)	4.578 9	4.565 8	4.546 7

注:视觉质量等级由高到低依次为 5,4,3,2,1。

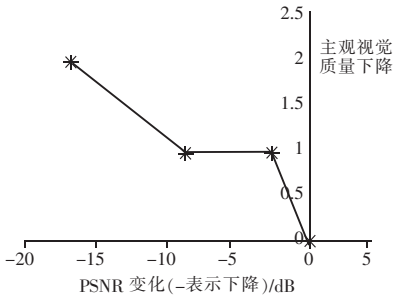


图 7　主观质量与 PSNR 变化关系

4　结束语

本文研究了在 H.264 的标准软件 JM 的编码部分植入由变异算子生成的简单错误对编码器效率的影响。本文的实验结果中,除了帧内预测模块,编码器中的绝大部分简单错误对编码的 PSNR 的影响很小,并且除少数情况,对视频的视觉质量影响也不明显,由此说明通过客观和主观质量的明显变化来确定编码器的错误是存在一定难度的,需要更进一步的研究。当然,由于水平和实验的范围所限,结论不一定非常全面。本文只是抛砖引玉,对编码器的测试还需要更进一步的研究。

参考文献:

[1] KIM C M, LEE B U, PARK R H. Design of mpeg-2 video test bitstreams[J]. IEEE Trans on Consumer Electronics, 1999, 45(4): 1213-1220.

[2] HARROLD M J, OFFUTT A J, TEWARY K. An approach to fault modeling and fault seeding using the program dependence graph[J]. Journal of Systems and Software, 1997, 3(3): 273-295.

[3] ANDREWS J H, BRIAND L C, LABICHE Y, *et al*. Using mutation analysis for assessing and comparing testing coverage criteria[J]. IEEE Trans on Software Engineering, 2006, 32(8): 608-624.

[4] OFFUTT A J. Investigations of the software testing coupling effect[J]. ACM Trans on Software Engineering and Methodology, 1992, 1(1): 5-20.

[5] DeMILLO R A, LIPTON R J, SAYWARD F G. Hints on test data selection help for the practicing programmer[J]. Computer, 1978, 11(4): 34-41.

[6] MA Y S, OFFUTT J, KWON Y R. An automated class mutation system [J]. Journal of Software Testing, Verification and Reliability, 2005, 15(2): 97-133.

[7] UMAR M. An evaluation of mutation operators for equivalent mutants [R]. London: King's College London, 2006.

[8] SUHRING K. JVT H.264/AVC reference software JM 18.0 [EB/OL]. http://iphome.hhi.de/suehring/tml/.