基于 H. 264 视频编码的运动估计算法优化

吴晓军,白世军,卢文涛
(哈尔滨工业大学深圳研究生院,广东深圳 518055)

摘 要: 运动估计是视频压缩中最重要的环节. H. 264 编码器由于采用了高精度运动矢量,计算量迅速增长,运动估计消耗整个编码时间 80 %左右. 本文在分析 UMHexagonS 算法的基础上,分别对 UMHexagonS 算法中搜索窗口大小的选择、大六边形搜索和小六边形(小钻石)搜索模式三方面做了优化,在保持了原有图象质量的情况下有效的节省了运动估计时间. 通过对各种测试序列的实验证明,优化后的算法与 UMHexagonS 算法相比,在重建图象质量和码率接近的情况下,运动估计时间平均节省了 18. 292 %,降低了算法的复杂度,提高了编码器的实时性.

关键词: 运动估计; 动态搜索窗口; UMHexagonS; H. 264
中图分类号: TN943 **文献标识码**: A **文章编号**: 0372-2112 (2009) 11-2541-05

Optimization on Motion Estimation Algorithm Based on H. 264

WU Xiao-jun, BAI Shi-jun, LU Wen-tao
(Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, Guangdong 518055, China)

Abstract: Motion estimation (ME) is vital to video compression. Due to the adoption of the high precision of motion vector (MV) in H. 264 encoder, the computational cost increases rapidly, and ME takes almost 80 % encoding time. In this paper, based on the UMHexagonS algorithm, an optimized algorithm is proposed based on the dynamic search range selection, big hexagon and small hexagon search mode respectively, which saves ME time effectively without quality loss. Experimenting with some typical video sequences proves that, compared to UMHexagonS algorithm, our optimized algorithm can save about 18. 292 % ME time and reduce the complexity of original scheme as well as enhance the real-time performance of the encoder but almost has no change in the reconstructed picture quality and bitrate.

Key words: motion estimation; dynamic search range; UMHexagonS; H. 264

1 引言

H. 264 相对于已有的视频编码标准[1],比如 H. 261/ H. 263 和 MPEG-1/ 2/ 4,采用了更加高效和精确的运动估计预测方法,尽管如此,其运算复杂度还是相当的高,难以实现实时编码的要求.

运动估计算法是视频压缩领域一个非常重要的研究热点,运动估计所要花费的时间占据了整个编码过程中相当大的比重,各国学者提出了诸多快速算法以减少运动估计的计算复杂度,同时基本保持了原有视频质量. 一些较好的快速搜索算法有新三步法(TSS)、四步法(FSS)[2]、六边形搜索(HS)[3]、钻石搜索[4]. 由于这些算法在不同程度上容易陷入局部最优,对小的运动视频序列效果比较好,而对大运动视频序列不是很理想.

为了解决过早的陷入局部最优,H. 264 参考模型 JM 采用了 UMHexagonS[5]算法. UMHexagonS 算法使用了混合扩展的运动搜索方法表现出了良好的编码效果,但在对某些块进行搜索匹配过程中还是没有很好的避免落入局部最优.

本文基于 UMHexagonS 算法,使用了动态搜索窗口,自适应大六边形和小六边形搜索模式,避免了搜索过程限于局部最优,在没有增加码率和保证图象质量的情况下,有效的节省了运动估计时间,降低了算法的复杂度,提高了编码器的实时性.

2 H. 264 中 UMHexagonS 算法

在以前编码标准中,运动估计(motion estimation)的块大小一般是 16 ×16,运动矢量(motion vector)用来表示块的相对位移. 为了能够更好的预测和获得更高的编码效率,H. 264 在运动估计中采用了不同大小的块进行预测编码. H. 264 编码支持的块大小有 16 ×16,16 ×8,8 ×16,8 ×8,8 ×4,4 ×8 和 4 ×4.

收稿日期:2008-09-19;修回日期:2009-01-12
基金项目:国家自然科学基金(No. 50805031);深圳市科技科计划项目(No. JC200903120184A, No. ZYC200903230062A)

UMHexagonS 之所以叫做非对称十字型多层次六边形格点运动搜索算法,是因为它包含四种搜索模式:即不对称的十字交叉搜索;非均匀多层次六边形格点搜索;六边形搜索;小钻石搜索. 算法流程如图 1 所示.

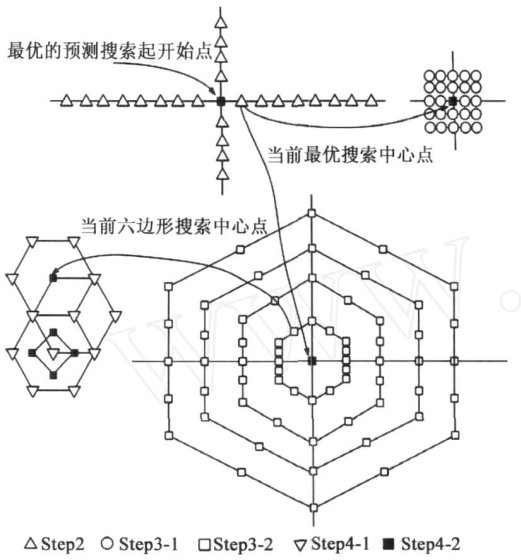

图1 UMHexagonS算法

UMHexagonS 算法中采用了固定大小的参考搜索窗口,通过配置文件由参数 *search_range* 来设置,*search_range* = 16/32/48/64. 在 H.264 标准中有 7 种不同的分割大小,最大块为 16 ×16,最小块为 4 ×4,参考搜索窗口的大小是由参数 *search_range* 决定,大小为 $(2^{*}search\_range+1)*(2^{*}search\_range+1)$,总共搜索的点数(*SP_number*)由式(1)给出:

$$SP\_number=(2^{*}search\_range+1)^{2} \quad (1)$$

搜索点数的多少是评价算法运算复杂度的一个指标,除了用各种快速算法来减少搜索点数外,7 种不同大小的块都在固定大小的参考窗口中搜索最佳匹配块是不科学的,7 种块大小不同而且运动矢量大小也不一样,比如对 4 ×4 小块势必会增加一些无用点的搜索;而对 16 ×16 的大块可能由于运动比较剧烈而在固定大小的窗口中无法找到最佳匹配的块,我们把这个标记为问题一.

图 1 所示 UMHexagonS 算法的主要部分使用了六边形搜索算法(HEXBS)[6],包括非均匀多层次六边形格点搜索和扩展的六边形搜索,非均匀多层次六边形格点的搜索点数为 $N1=16*4=64$,扩展的六边形搜索如图 2 所示,点数 $N2=7+3*n+4$. 六边形搜索算法虽然较大钻石[4]减少了搜索点数,降低了算法复杂度,但是在每次新预测矢量(PMV)确定后,都要排除新六边形中已经被搜索过的三个点,同时要检查没有被搜索过的其他三个点,这样势必会增加软件开销,有待进一步优化,我们把这个标记为问题二.

UMHexagonS 算法最后一步在六边形搜索基础上用了小钻石搜索算法,如图 3 所示来检查点 2、4、5、7 四个点,这样就漏掉了对点 1、3、6、8 的检测,用全搜索检测八个点,又会增加算法复杂度,此处也有待改进,我们把它标记为问题三.

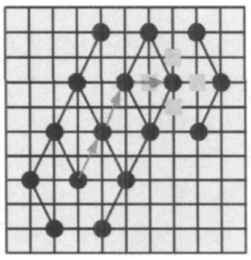

图2 六边形搜索模式

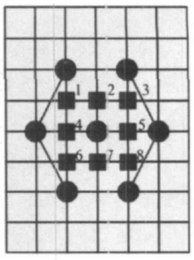

图3 六边形内部点分布

3 优化后的 UMHexagonS 算法

3.1 动态搜索窗口的引入(AP1)

针对在第二部分分析中所提出的问题一,我们利用动态搜索窗口代替固定搜索窗口[7],动态搜索窗口更有利于降低算法复杂度和提高估计的精度,针对不同大小的当前块每次动态新生成搜索窗口,把这部分优化称为 AP1,AP1 中用运动矢量的中值预测值(*MVPmedian*)和上层预测值(*MVPuplayer*)来计算动态搜索窗口(DSR)大小.

动态搜索窗口(DSR)计算过程如图 4 所示,其中:

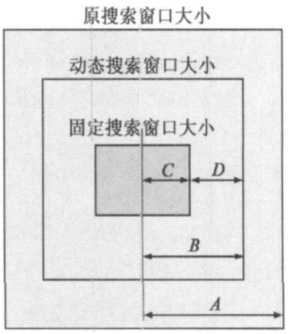

图4 AP1动态搜索窗口(DSR)的计算

A:原算法输入的搜索窗口大小,即 *input_search_range*;

B:动态搜索窗口的大小,由 C 和 D 构成;

C:通过大量的实验并分析实验结果得到的经验值:*fixed_part* = (*input_search_range*)/8;

D:*dinamic_part* 由式(3)计算,$MVPuplayer_x$,$MVPuplayer_y$ 和 $MVPmedian_x$,$MVPmedian_y$ 分别表示 *MVPuplayer* 和 *MVPmedian* 的 *x*,*y* 分量.

动态搜索窗口大小 B 由式(2)计算:

$$prpsd\_DSR=fixed\_part+dinamic\_part \quad (2)$$

$$dinamic\_part=\max(mvxd,mvyd) \quad (3)$$

$$mvxd=abs(MVPuplayer_x-MVPmedian_x) \quad (4)$$

$$mvyd=abs(MVPuplayer_y-MVPmedian_y) \quad (5)$$

在式(2)中,*prpsd_DSR*,*fixed_part*,*dinamic_part* 分别表示 AP1 中动态搜索窗口(DSR)的大小、DSR 中固定

部分大小和 DSR 中动态部分大小. 有一点需要补充说明的是,如果当前块是 16 ×16,那么不存在 *MVPuplayer*,此时搜索窗口的大小按文献[8]的方法计算.

3.2 改进的六边形搜索模式(AP2)

针对在第二部分分析中所提出的问题二,我们利用当前块运动矢量(*current_mv*)和运动矢量(*col_mv*)来减少六边形搜索点数[9],其中 *col_mv* 表示前一帧与当前块相同坐标位置的块的运动矢量,把这部分优化称为 AP2.

如果 *current_mv* 和 *col_mv* 都不等于 0,则计算出两个矢量的夹角 α,否则按原算法进行处理,一旦当前块的位置确定后,*current_mv* 的大小就确定了,*col_mv* 在一次搜索过程大小是不变化的,而 *current_mv* 的大小随着搜索中心点的改变而改变. 具体 AP2 搜索方法的步骤由 Step0、Step1、Step2、Step3 详细描述如下:

Step0　UMHexagonS 算法中,扩展多层次六边形十六格点搜索如图 5(*a*)所示,以相对坐标原点(0,0)为中心,需要搜索搜索的点数是 4 ×16 = 64 点,在这里我们根据运动矢量 *current_mv* 和 *col_mv* 的夹角 α 来决定计算如图 5(*a*)、(*b*)、(*c*)、(*d*)中的一个,搜索点数为 4 ×5 = 20,这样大大降低了搜索点的数.

Step1　以 Step0 中计算的最优点如图 5(*f*)(0,0)为中心搜索周围六个点,如果中心点(0,0)就是最优点,那么直接跳转到 Step3 (Ending);否则,计算 *current_mv* 和 *col_mv* 的夹角 α.

Step1(a**)**　if α = 0 to 90 则搜索图 5(*g*)所示的三个点,在这三个点中找到最优点后跳转到 Step2(搜索).

Step1(b**)**　if α = 90 to 180 则搜索图 5(*h*)所示的三个点,在这三个点中找到最优点后跳转到 Step2(搜索).

Step1(c**)**　if α = 180 to 270 则搜索图 5(*i*)所示的三个点,在这三个点中找到最优点后跳转到 Step2(搜索).

Step1(d**)**　if α = 270 to 360 则搜索图 5(*j*)所示的三个点,在这三个点中找到最优点后跳转到 Step2(搜索).

Step2　以上步搜索得到最优点为中心,构造新六边形,检查两个新的后选点,如果最优点是新六边形中心点,直接跳转到 Step3 (Ending),否则,跳转到 Step1 按图 5 重复上面的步骤计算.

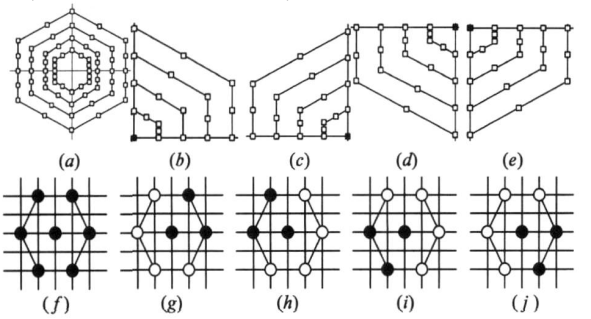

图5　AP2搜索模式 (*b*)(*g*) if α= 0 to 90, (*c*)(*h*) if α= 90 to 180 (*d*) (*i*) if α=180 to 270 (*e*) (*j*) if α=270 to 360

Step3　以最优点为中心搜索小六边形的两个点,这两个点的位置由 α 的大小决定,其中的最优点就是整个搜索的最优点,结束搜索.

3.3 六边形内部搜索模式(AP3)

针对在第二部分分析中所提出的问题三,如图 2 所示在六边形搜索最后一步,由大六边形搜索模式转为小六边形(小钻石)搜索模式时,分别检查了搜索中心点周围的四个点. 如果 *current_mv* 和 *col_mv* 其中一个为 0 或者相等,那么按原 UMHexagonS 算法处理,否则根据文献[10],对此搜索模式进一步优化,把这部分优化称为 AP3.

在六边形搜索中,第二阶段的内部搜索(inner search)须检查最小值中心点周围四个点,而在[10]中利用了在全局最小周围的单调失真特性(monotonic distortion characteristic),使得内部点搜索(inner search)只需要搜索一个特定方向的内部点(inner points). 如图 6 所示,AP3 额外计算了周围六个 group 两个块的 SAD 之和. 如图 6(*a*)所示,假如最小 SAD 和是 Group2,只需搜索 3 个点,如图 6(*b*)所示,如果是 Group1,则只需搜索 2 个点. 因此在搜索速度和失真的表现上要比六边形搜索模式好.

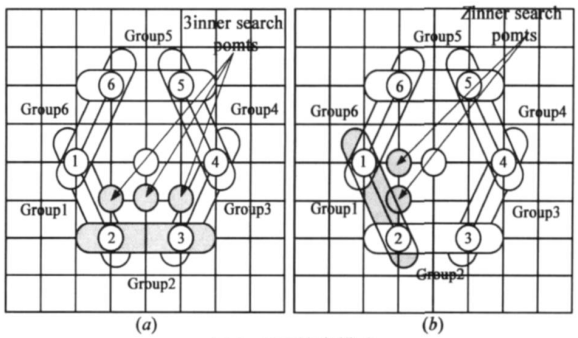

图6　AP3搜索模式

由于六边形搜索算法最后一步是搜寻中心点周围四个点,而 AP3 中根据 Group 分组分别减少 1 或者 2 个搜寻点,所以大约减少了 2 * (4/6) + 1 * (2/6) = 1.67 个点,因为是每个搜索块都减少了大约 1.67 个点,所以一个 frame 就减少了相当多的搜寻点,自然其速度比六边形搜索算法快很多. 而整个搜索算法的搜索总点数为 7 + 3 * *n* + (2,3).

4 实验结果和性能分析

4.1 测试平台及其配置

首先在 VC6.0 的平台上将改进算法用 C 语言实现,并将其集成到 H.264 标准测试软件 JM10.1 中. 实验所用 PC 机的硬件配置如下:Intel(R) Pentium(R) D CPU 3.00GHz,512M 内存. 操作系统为 WindowsXP 2002 + SP2.

测试序列集为 5 个 QCIF(176 *144) 格式序列所有序列都为 Yuv4 2 0. 编码器配置选用 JM10.1 的基本类(编码配置文件为: *encoder_ baseline. cfg*). 实验中主要编码参数如下: *FramesToBeEneoded* = 100, *FrameRate* = 30.0, *UseHadamard* = *l*, *SearehRange* = 16, *NumberRefereneeFrames* = 5 其他参数为缺省设置.

4.2 测试结果与分析

测试中 UMHS 表示原算法 UMHexagonS, AP123 表示把三个点的优化 AP1、AP2 和 AP3 同时加到原 UMHexagonS 算法中的意思. 选择了 5 个含有不同特点运动类型的标准测试序列进行实验, 实验数据见表 1. 从统计结果可以发现, 改进后的算法比原算法的编码时间平均节省了 10.528 %, 运动估计时间平均节省了 18.292 %, 而 *PSNR* 提高了 0.01dB 或者最大下降了 0.02dB 基本保持了原有视频质量. 另外还发现 AP3 对 Highway 等复杂运动序列的的改进效果比其它简单运动序列的改进效果要好得多.

表 1 测试结果比较

测试序列	算法版本	*PSNR*(dB)	码率(kbit/s)	Enc. T 节省率(%)	ME. T 节省率(%)
Mobile	UMHS	33.35	425.87	0.00	0.00
	AP1	33.34	426.05	8.58	20.76
	AP2	33.35	425.82	5.68	8.84
	AP3	33.36	425.81	-1.50	-1.48
	AP123	33.36	426.45	17.29	28.23
Coastguard	UMHS	34.04	248.97	0.00	0.00
	AP1	34.05	248.70	9.48	19.38
	AP2	34.04	248.08	2.03	5.59
	AP3	34.05	249.37	-3.37	-5.29
	AP123	34.05	249.55	13.24	22.76
News	UMHS	36.70	75.05	0.00	0.00
	AP1	36.72	75.24	2.72	10.76
	AP2	36.72	75.37	7.94	10.19
	AP3	36.70	75.07	-3.09	-1.10
	AP123	36.68	75.26	6.86	13.13
Foreman	UMHS	36.47	135.11	0.00	0.00
	AP1	36.48	135.54	-0.88	7.65
	AP2	36.49	135.13	3.80	5.22
	AP3	36.47	135.16	-6.21	-5.83
	AP123	36.49	135.85	3.65	11.80
Highway	UMHS	37.78	68.54	0.00	0.00
	AP1	37.79	69.25	9.71	13.53
	AP2	37.78	70.56	8.24	11.29
	AP3	37.79	68.57	6.02	6.65
	AP123	37.78	70.73	11.60	15.54

从表 2 可以看出, 本文优化后的算法 AP123 针对不同的运动序列都能保持优异的压缩性能, AP123 算法具有以下特点: (1) 与 FS 相比, 亮度信号的 *PSNR* 性能平均提高了 0.01dB, 最大损失不超过 0.03dB, 与 EPZS 算法相比, 亮度信号的 *PSNR* 性能平均损失了 0.01dB, 最大损失不超过 0.06dB, 视频的重建质量基本保持了原有图象质量; (2) 与 FS 相比, 比特率的增加很少, 平均值为 2.4946 %, 与 EPZS 相比, 平均值为 0.736 %, 编码的效率基本不变; (3) 运动估计和编码的耗时明显下降, 与 FS 相比, 运动估计速度约为 FS 的 3.352 倍, 与 EPZS 相比, 运动估计时间平均下降了 12.786 %. AP123 与 FS 和 EPZS 相比, 在重建图象质量和码率接近的情况下, 实时性有了明显的提高.

表 2 三种搜索算法测试结果比较

测试序列	算法版本	*PSNR*(dB)	码率(kbit/s)	Enc. Time	ME. Time
Mobile	FS	33.35	423.26	255.002	179.775
	EPZS	33.36	424.79	131.464	63.994
	AP123	33.36	426.45	121.972	53.602
Coastguard	FS	34.01	244.86	247.372	185.607
	EPZS	34.05	249.10	125.396	69.758
	AP123	34.05	249.55	114.248	59.012
News	FS	36.71	75.27	235.594	178.730
	EPZS	36.74	75.15	104.393	52.618
	AP123	36.68	75.26	96.917	45.603
Foreman	FS	36.46	132.29	229.982	175.673
	EPZS	36.50	135.57	116.107	65.665
	AP123	36.49	135.85	115.764	64.815
Highway	FS	37.78	66.03	225.393	177.396
	EPZS	37.80	68.52	103.842	58.953
	AP123	37.78	70.73	93.889	48.538

5 结论

H.264 在有效提高编码效率的同时, 使运动估计模块的计算复杂度随之成倍增加, 极大地增加了 H.264 的计算量. 本文对 H.264 中运动估计 UMHexagonS 算法进行了分析, 用动态自适应搜索窗口代替了原算法固定搜索窗口, 在进行大六边形和小六边形搜索时锁定了搜索方向, 有效降低了搜索点数. 从实验结果可以看出, 本文算法在保证编码性能的情况下, 可以有效地降低 H.264 运动估计过程的复杂度. 在编码时间和运动估计时间上都不同程度的有所减少, 特别是对运动较大序列效果更加明显, 提高了编码器的实时性. 因此, 本文提出的改进算法是一种适合 H.264 的速度快而性能好的运动估计算法, 与现有大多算法相比, 具有一定的优势.

参考文献:

[1] Wiegand T. Sullivan G. J. Luthra A. Overview of the H.264/AVC video coding standard[J]. IEEE Transactions on Circuits and System for Video Technology, 2003, 13(7): 560 - 576.

[2] Po L. M. Ma W. C. A novel four-search algorithm for block motion estimation[J]. IEEE Transactions on Circuits and Systems for Video Technology, 1996, 6(3): 313 - 317.

[3] Zhu C. Lin X. Chau L. P. Hexagon based search pattern for fast block motion estimation[J]. IEEE Transactions on Circuits and System for Video Technology, 2002, 12(5): 349 - 355.

[4] Tham J. Y. Ranganath S. Kassim A. A. A novel unrestricted center-biased diamond search algorithm for block motion estimation[J]. IEEE Transactions on Circuits and Systems for Video Technology, 1998, 8(4): 369 - 377.

[5] Chen Z. Xu J. He Y. Zheng J. Fast integer-pel and fractional-pel motion estimation for H.264/AVC[J]. Journal of Visual Communication and Image Representation, Journal of Visual Communication and Image Representation, 2006, 17(2): 264 - 290.

[6] Zhu C. Lin X. Chau L.-P. Hexagon-based search pattern for fast block motion estimation[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2002, 12(5): 345 - 355.

[7] Xu X. Z. He Y. Modification of dynamic search range for JVT[S]. JVT-Q088, Virginia, USA, 2002.

[8] Chen Z. X, Song Y. Ikenaga T. Goto S. A dynamic search range algorithm for variable block size motion estimation in H.264/AVC information[A]. the 6th International Conference on Communication & Signal Processing[C]. Singapore, 10 - 13 Dec, 2007. 1 - 4.

[9] Thambidurai P. Ezhilarasan M. Ramachandran D. Efficient motion estimation algorithm for advanced video coding[A]. Conference on Computational Intelligence and Multimedia Application[C]. Sivakasi, Tamilnadu, India, 2007. 47 - 52.

[10] Zhu C. Lin X. Chau L. PO L. M. Enhanced hexagonal search for fast block motion estimation[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2004, 14(10): 1210 - 1214.

作者简介:

吴晓军　男, 1975 年生于甘肃张掖,哈尔滨工业大学深圳研究生院副教授,2004 年获得中国科学院沈阳自动化研究所机械电子工程博士后学位,主要从事计算机视觉、数字图像处理、计算机图形学及曲面三维重建等.

白世军　男,1981 年 6 月出生于甘肃宁县,2006 年考入哈尔滨工业大学深圳研究生院. 现为哈尔滨工业大学深圳研究生院控制科学与工程专业硕士研究生,主要从事数字视频处理技术研究.

E-mail: baishijun625@yahoo.com.cn