第35卷 第24期 计 算 机 工 程 2009年12月
Vol.35 No.24 Computer Engineering December 2009
·多媒体技术及应用· 文章编号：1000—3428(2009)24—0229—03 文献标识码：A 中图分类号：TN919.8

自适应的动态搜索范围运动估计算法

袁 涛，汪同庆
(重庆大学光电工程学院人工视觉研究实验室，重庆 400030)

摘 要：为更加有效地提高运动估计速度，提出一种自适应动态搜索范围运动估计算法，从后续快速运动估计算法的运动矢量预测集中自适应地选择与当前编码块相关性最强的运动矢量预测值作为搜索范围的中心点，根据预测集中运动矢量预测值的大小、方向自适应地决定水平、垂直及正负方向的非对称搜索范围。将该算法融合到UMHexagonS 和 FFS 算法中进行广泛的实验测试，结果表明其能在基本保持重建图像质量的同时，至少分别减少运动估计运算量的22.13%和76.57%。

关键词：视频编码；运动估计；运动矢量；动态搜索范围

Adaptive Dynamic Search Range Algorithm for Motion Estimation

YUAN Tao, WANG Tong-qing
(Artificial Vision Research Laboratory, College of Opto-electronic Engineering, Chongqing University, Chongqing 400030)

【Abstract】 To accelerate the motion search process, this paper proposes a novel adaptive dynamic search range algorithm. It selectes dynamically the predictive Motion Vector(MV), whose relevant referenced block is most correlative with the current block, in candidate MV set of backward fast motion estimation algorithm as the center of search range. It determines adaptively the unsymmetrical search range in negative/positive direction of horizontal/vertical axis according to the predictive MV set to reduce the search points. Extensive experiments show that this proposed scheme, combined with UMHexagonS and FFS, can achieve more than 22.13% and 76.57% computational reduction of motion estimation respectively, compared with the previous algorithms, while maintaining almost the same quality of reconstructed pictures.

【Key words】 vedio coding; motion estimation; Motion Vector(MV); dynamic search range

1 概述

在H.264中，由于采用了1/4像素精度预测、灵活的分块策略、多参考帧预测等技术来对编码图像进行更为准确的预测，从而有效地减少了视频图像的比特率，但与此同时也显著增加了整个编码过程的运算复杂度。相对于H.263和MPEG4，H.264能在获得相同编码图像质量的同时节省大约50%的码率，但是编码、解码过程运算量相对MPEG4则分别提高了大约9倍和4倍，而其中运动估计在整个编码过程中占据了相当比重的运算量。在采用单参考帧和多参考帧预测的情况下，运动估计的运算量分别占整个编码过程运算量的60%和 80%左右。因此，通过减少运动估计运算量来加快整个编码过程的速度就显得特别重要。

在快速运动估计算法中，一般通过2种途径来减少运动估计的运算量：

(1)动态搜索范围快速算法，这类搜索算法主要是利用编码序列时间、空间相关性动态调整运动估计的搜索范围大小，从而减少搜索点数以加快运动估计的速度。

(2)固定模式的快速运动估计搜索算法，如三步法(3SS)、钻石搜索算法(Diamond)、非对称十字型多层次六边形格点搜索算法(UMHexagonS)，这类算法一般分为2步：粗精度搜索和细精度搜索，粗精度搜索是先通过加大搜索步长，迅速将搜索范围锁定在最佳匹配点可能出现的位置附近，再通过细精度搜索算法来进一步精细最佳匹配位置，从而得出当前编码块的运动矢量(Motion Vector, MV)。典型的如钻石搜索算法，先通过大钻石搜索模式迅速定位到最佳匹配位置附近，再通过小钻石模式进行精确定位从而得出最后的MV。

2 动态搜索范围算法

在JM模型中，是通过固定的输入值input_search_range来控制运动估计的搜索范围，当搜索范围设置较大时，可以得到比较理想的运动估计效果，获得较好的重建图像质量，但同时也增加了运动估计的运算量；而当减少搜索范围时，虽然可以减少运动估计的运算量，但却降低了重建图像的质量。而动态搜索范围算法则是根据当前编码块的纹理特征以及利用时间、空间相邻已编码块来预测当前编码块的运动矢量，并据此动态调整搜索范围，从而在保证编码图像质量的同时有效地减少了运动估计的运算量。文献[1]提出了一种利用视频图像帧间统计特性及运动向量时域、空域相关特性的自适应搜索范围算法，由于该搜索算法基于每一帧通过对前一帧运动矢量的统计特性来调整搜索范围，而运动估计是基于块来进行的，显然当帧中存在运动矢量小的静态、准静态部分和运动矢量较大的复杂、剧烈运动部分时，无法准确确定当前编码块的搜索范围，并且对前一帧所有运动矢量的统计需要较多的存储空间来存储前一帧的所有运动矢量，这需要耗费较大的运算量。文献[2]提出根据当前编码宏块空间相邻4×4块(A,B,C)MV的所有x,y分量的最大绝对值的2倍来确定搜索范围，这种算法是基于编码宏块来计算运动搜索范围的，很好地解决了文献[1]中由于运动搜索范围在帧的基础

基金项目：国家科技支撑计划基金资助项目(2007BAG06B06)
作者简介：袁 涛(1983－)，男，硕士研究生，主研方向：基于H.264的运动估计算法；汪同庆，教授、博士生导师
收稿日期：2009-07-14 **E-mail：**ytcqu@163.com

上抉择而运动估计在块的基础上进行所引起的不匹配问题，同时也不需要过多的存储空间和运算量。但其对 *x,y* 以及正负 4 个方向设置相同的搜索范围，而对于特定的编码块来说，4 个方向的运动情况并不完全相同。文献[3]建立在文献[2]的基础上，*x,y* 方向的搜索范围分别根据当前编码块相邻 4×4 块(A,B,C)MV 在 *x,y* 分量上最大绝对值的 2 倍单独决定，显然相对于文献[2]中的算法，该算法可以更加有效地缩小运动估计的范围。在文献[4]中，针对相邻 4×4 块的运动矢量较小时可能导致最后搜索范围过小的情况设置了搜索范围下限，即根据 A,B,C 的 *x,y* 分量绝对值之和的大小来设置搜索范围的下限。

3 自适应的动态搜索范围

优秀的动态搜索范围算法就是要使搜索范围能最大限度地覆盖全局最佳匹配点以得到与全搜索算法相同的重建图像质量，同时又能有效地减少搜索的范围。决定搜索范围的因素有 2 个：搜索范围中心点和搜索范围大小。本文所提出的 ADSR 算法将在搜索范围中心点和搜索范围大小上做出改进，以得到优于现有算法的效果。

3.1 搜索范围中心的确定

3.1.1 与 UMHexagonS 结合时 ADSR 算法搜索中心的确定

相对于全搜索算法，UMHexagonS[5]能减少 90%的运算量，同时能得到几乎相同的重建图像质量，已被 JVT 正式采纳为 H.264 的运动估计快速算法。UMHexagonS 采用了更为精确的初始点预测技术来预测当前编码块 MV 的最佳可能点作为其初始搜索点，从而极大地提高了运动估计的速度和避免了陷入“局部最佳点”的问题。

考虑到 ADSR 算法将作为快速运动估计算法的预处理部分，且 UMHexagonS 的初始搜索点预测技术已由实验证明具有良好的效果，当应用此方法来确定搜索范围中心点时也不需要增加额外的运算量和存储器开销，所以在 ADSR 算法中，采用 UMHexagonS 的初始搜索点预测技术来预测全局最佳匹配点最可能出现的位置作为搜索范围的搜索中心点。

在 UMHexagonS 中，采用了 4 种 MV 预测模式：空间中值预测 $\boldsymbol{MV}_{pred\_MP}$，上层子块模式预测 $\boldsymbol{MV}_{pred\_UP}$，前帧对应块预测 $\boldsymbol{MV}_{pred\_CP}$，相邻参考帧预测 $\boldsymbol{MV}_{pred\_NRP}$，由该 4 种模式组成初始预测点集合 S_1：

$$S_1=\{\boldsymbol{MV}_i \mid \boldsymbol{MV}_i = \boldsymbol{MV}_{pred\_MP},\boldsymbol{MV}_{pred\_UP},\boldsymbol{MV}_{pred\_CP},\boldsymbol{MV}_{pred\_NRP},(0,0)\} \quad (1)$$

实验表明，MV 预测集 S_1 附近位置与全局最佳匹配点也有很强的相关性，所以有必要包括这些向量：

$$\psi(\boldsymbol{CMV})=\{\boldsymbol{MV}_i \mid \boldsymbol{MV}_i = (CMV_x\pm1,CMV_y),(CMV_x,CMV_y\pm1)\} \quad (2)$$

同时定义另一个预测集 S_2：

$$S_2=\psi(\boldsymbol{MV}_{pred\_MP})\cup\psi((0,0)) \quad (3)$$

因此，总的 *MV* 预测集为 $S=S_1\cup S_2$。

最后通过计算预测集 *S* 中代价值最小的 *MV* 预测值作为初始搜索点，也就是动态搜索范围的中心点。

$$\boldsymbol{MV}_{center}=\arg[\min_{\boldsymbol{MV}_i} Cost(\boldsymbol{MV},\lambda MOTION)] \quad \text{s.t. } \boldsymbol{MV}_i\in S \quad (4)$$

代价值为当前编码块与参考块的 *SAD* 值和该 *MV* 预测值所需的比特率代价之和。

3.1.2 与 FFS 结合时 ADSR 算法搜索中心的确定

相对于全搜索算法，快速全搜索算法(FFS)能在基本保持重建图像质量的同时减少 20%的运动估计时间。在 FFS 中，采用空间中值预测点 $\boldsymbol{MV}_{pred\_MP}$ 作为运动估计的初始搜索点，所以，在与 FFS 算法结合时，ADSR 算法定义的 *MV* 预测集 *S* 为

$$\{(0,0),(\boldsymbol{MV}_{x\_Media},\boldsymbol{MV}_{y\_Media}),(\boldsymbol{MV}_{x\_a},\boldsymbol{MV}_{y\_a}),(\boldsymbol{MV}_{x\_b},\boldsymbol{MV}_{y\_b}),(\boldsymbol{MV}_{x\_c},\boldsymbol{MV}_{y\_c})\} \quad (5)$$

计算 *S* 中代价最小的 *MV* 预测值所指的位置作为搜索中心点。

在 FFS 中，是通过先计算每个编码块在 inter4×4 模式下的 *SAD* 值再通过组合的方式得出其他所有模式下的 *SAD* 值，然后将运动矢量代价 *MV_COST* 与其 *SAD* 值相加得出各个模式下的代价值。所以 *SAD* 的计算与 *MV_COST* 是分开进行的，因此，选择最佳可能匹配位置时，是以 *SAD* 值为依据来选择最有可能的最佳匹配点作为搜索范围中心点的。

3.2 ADSR 算法搜索范围的确定

搜索范围的设定是要最大概率地覆盖全局最佳点，同时又要使搜索窗尽可能小以减少搜索点数从而加快搜索速度。

当编码块运动复杂、剧烈时，根据各种模式对当前编码块进行预测的 *MV* 预测值的差别也较大；而当编码块为静止、运动量小的块时，各种模式 *MV* 预测值之间的差别也较小。基于以上原理，ADSR 算法提出一种依据 *MV* 预测集中 *MV* 预测值的大小、方向来对 *x,y* 正负 4 个方向分别进行预测从而更加有效地确定在每个方向上的非对称搜索距离，以便在保证质量的同时能最大限度地减少运动搜索的点数。

3.2.1 与 UMHexagonS 结合时 ADSR 算法搜索范围的确定

与 UMHexagonS 结合时 ADSR 算法搜索范围确定的步骤如下：

Step1 检查预测集 *S* 中 *MV* 预测值对于当前编码块是否可用，如果可用 *MV* 预测值数目大于 1，进入 Step2，否则转入 Step4。

Step2 分别得出 *MV* 预测集 *S* 中所有 *MV* 预测值在 *x,y* 分量上的最大值、最小值：

$$Max\_mv\_i=\max\{\boldsymbol{MV}_i\}\quad \boldsymbol{MV}\in S,\ i=x\text{或}y$$
$$Min\_mv\_i=\min\{\boldsymbol{MV}_i\}\quad \boldsymbol{MV}\in S,\ i=x\text{或}y \quad (6)$$

再依据之前得出的搜索范围中心点分别得出所有 ***MV*** 预测值在 *x,y* 正负方向上的变化范围：

$$Length\_i\_low=\boldsymbol{MV}_{center\_i}-Min\_mv\_i,\ \ i=x\text{或}y$$
$$Length\_i\_high=Max\_mv\_i-\boldsymbol{MV}_{center\_i},\ \ i=x\text{或}y \quad (7)$$

Step3 大量实验表明，当把搜索范围设为变化范围 2 倍时可以在减少搜索范围的同时基本保持重建图像的质量。为避免当 *MV* 预测集 *S* 中所有 *MV* 较小或接近零可能导致动态搜索范围过小而产生较大误差的情况，对搜索范围设置下限为 *input_search_rang*/8。

$$Length\_i\_low=\min(input\_search\_range,\max(input\_search\_rang/8,2\cdot Length\_i\_low)),i=x\text{或}y$$
$$Length\_i\_high=\min(input\_search\_rang,\max(input\_search\_rang/8,2\cdot Length\_i\_high)),i=x\text{或}y \quad (8)$$

所以，最后得出在 *i*(*i*=*x* 或 *y*)轴方向上的搜索范围为

$$(\boldsymbol{MV}_{center\_i}-Length\_i\_low,\boldsymbol{MV}_{center\_i}+Length\_i\_high)\quad i=x\text{或}y \quad (9)$$

Step4 如果集合 *S* 中可用的 *MV* 只有一个，则仍将搜索范围设为默认的 *input_search_rang* 不变。

3.2.2　与 FFS 结合时 ADSR 算法搜索范围的确定

与 FFS 结合时 ADSR 算法搜索范围确定所采用的策略同其与 UMHexagonS 算法结合时一样，只是 *MV* 预测集 *S* 不同：

$$\{(0,0),(\boldsymbol{MV}_{x\_Media},\boldsymbol{MV}_{y\_Media}),(\boldsymbol{MV}_{x\_a},\boldsymbol{MV}_{y\_a}),(\boldsymbol{MV}_{x\_b},\boldsymbol{MV}_{y\_b}),(\boldsymbol{MV}_{x\_c},\boldsymbol{MV}_{y\_c})\} \quad (10)$$

4　仿真结果与分析

实验是在 JVT 的 H.264 编解码参考模型 JM8.6 上实现的，选择具有广泛代表性的 8 个测试序列进行测试，CIF(352×288)序列：stefan, bus, coastguard，QCIF(176×144)序列：foreman, salesman, suize, news, silent。其中，stefan, bus, coastguard 为大运动序列；foreman, salesman, silent 为中等运动序列；suize, news 为小运动序列。foreman 中包含了镜头转换。对 CIF, QCIF 的搜索范围分别设置为 16, 32，采用 Hadamard 变换、CABAC 熵编码和 5 个参考帧，编码后的序列，除首帧外，其余各帧均编码为 P 帧，即采用 IPPP…的方式，率失真优化，量化系数为 28；实验机型为 P4 Dual 2.0 GHz，内存 1 GB。

在实验中，编码、运动估计运算量用所需运算时间来评估，重建图像质量用 *PSNR_Y* 值来评估，比特率用每帧图像所需的平均比特数来度量。

图 1 给出了将 ADSR 算法与 FFS 和 UMHexagonS 算法结合前后的性能对比。其中，损失的 *PSNR_Y* 为加入 ADSR 算法后重建图像 *Y* 分量与没有加入该算法之前 *PSNR* 值的差值，增加 *Bit_rate*，减少 *Time*，减少 *ME_Time* 分别为加入 ADSR 算法前后的比特率、整体编码时间、运动估计时间变化百分比。

		QCIF					CIF		
		forman	suzie	news	salesman	silent	stefan	bus	coastguard
帧数		300	100	250	200	300	90	149	200
与 UMHexagonS 结合	损失 *PSNR_Y*/dB	0.02	-0.01	0.00	0.01	0.01	0.01	0.00	0.01
	增加 *Bit_rate*/(%)	0.25	-0.34	0.08	0.10	0.54	-0.13	0.13	0.20
	减少 *Time*/(%)	2.98	2.66	1.46	1.29	1.68	5.06	5.60	5.40
	减少 *ME_Time*/(%)	25.25	24.18	26.70	23.60	22.13	31.60	29.60	27.70
与 FFS 结合	损失 *PSNR_Y*/(%)	0.01	0.02	0.03	0.01	0.01	0.03	0.02	0.01
	增加 *Bit_rate*/(%)	5.13	1.30	1.20	-0.10	3.48	2.30	0.56	0.60
	减少 *Time*/(%)	31.63	33.76	32.90	32.00	31.87	54.80	53.96	56.25
	减少 *ME_Time*/(%)	78.00	80.00	78.80	79.10	79.03	79.40	76.57	80.50

图 1　加入本文所提出的动态搜索范围算法前后效果对比

由图 1 可知，对于 UMHexagonS 算法，在加入 ADSR 前后编码所需的比特率基本不变，*PSNR_Y* 的损失平均为 0.01 dB，最大为 0.02 dB，其损失远低于运动估计 *PSNR* 损失上限 0.1 dB，同时显著提高了运动估计的时间和整个编码时间；而对于 FFS 算法，就个别序列而言，*PSNR_Y* 值降低得稍微多一点，比特率也有一些增加，但是仍在准许的范围。造成这种差别的主要原因在于 UMHexagonS 算法中采用了比较严格、有效的 ***MV*** 预测，而 FFS 算法中仅仅利用了与当前编码块空间相邻的编码块来预测。

对于 UMHexagonS 快速算法，本文算法可以减少运动估计运算量的 22.13%~31.6%，由此可分别将 QCIF 和 CIF 的整个编码时间减少 1.29%~2.98%, 5.06%~5.6%，造成差别的原因是在 QCIF 序列中运动估计时间在整个编码时间中所占比例要远低于 CIF 序列中所占的比例。对于 FFS 算法，本文算法可以减少运动估计时间的 78%~80.5%，由此可分别将 QCIF 和 CIF 的整个编码时间减少 31.63%~33.76%, 53.96%~56.25%。对于 FFS 算法来说，由于运动估计在整个编码过程中所占时间约为 40%(QCIF), 70%(CIF)，因此加入 ADSR 后能将整个编码所需时间显著减少。

图 2、图 3 分别为 UMHexagonS, FFS 算法及加入 ADSR 算法后 foreman 测试序列每一帧图像的 *PSNR_Y* 值和运动估计时间的前后对比。由图 2 可知，不管是对于 FFS 还是 UMHexagonS 来说，加入 ADSR 后每一帧图像编码质量都能得到很好的保持，甚至在个别编码帧能取得比原来算法更好的图像质量，这主要是得益于 ADSR 算法采用最佳匹配点预测技术来预测搜索范围中心，能更好地将搜索范围锁定在最佳匹配点附近。由图 3 可知，对于 FFS 算法，加入 ADSR 算法后能显著地减少运动估计的时间，而对于 UMHexagonS 算法，加入 UDSR 算法后的效果不如 FFS 算法时明显，但是仍然能比较显著地减少运动估计的时间。

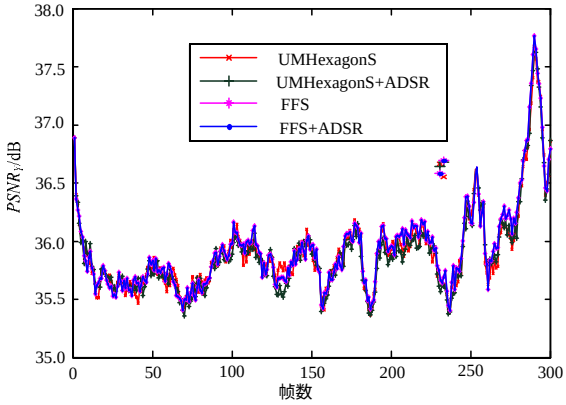

图 2　每一帧 *PSNR_Y* 对比(foreman 序列)

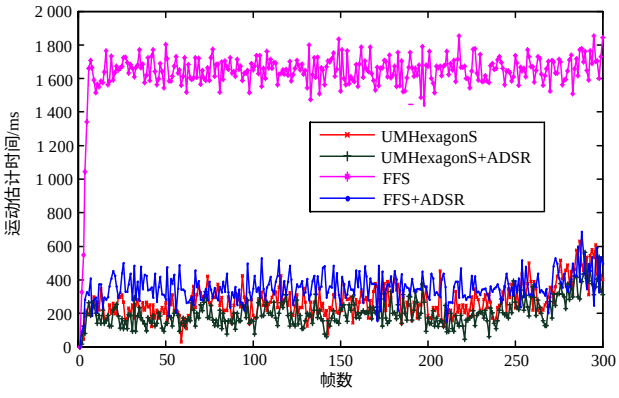

图 3　每一帧运动估计时间对比(foreman 序列)

(下转第 235 页)