视频数据存储方式及运动估计算法的内存访问效率

赵乘麟[1]，刘江云[2]，赵云辉[1]

(1. 邵阳学院 信息与激光研究所，湖南 邵阳，422000;
2. 邵阳学院 教育科学系，湖南 邵阳，422000)

摘 要：通过分析常用运动估计算法的内存访问模式，揭示出他们在通用计算机和数字信号处理器上使用时存在的效率问题，以及造成此问题的原因，并提出一种新的内存访问效率高的视频数据存储方式即叠瓦式存储方式。研究结果表明：使用新的视频数据存储方法，解决了跨缓存线访问数据的问题，并显著降低了运动估计过程中的高速缓存跑靶率；对全搜索算法，在算法本身不做任何改动的情况下，其运动估计过程速度提高 28%。因此，叠瓦式存储方式的使用，可以有效地加速运动估计过程。

关键词：视频数据；内存；访问效率；运动估计
中图分类号：TP391 **文献标识码**：A **文章编号**：1672-7207(2007)03−0540−05

Video data layout optimization and memory access analysis for motion estimation

ZHAO Cheng-lin[1], LIU Jiang-yun[2], ZHAO Yun-hui[1]

(1. Institute of Information & Laser, Shaoyang College, Shaoyang 422000, China;
2. Educational Science Department, Shaoyang College, Shaoyang 422000,China)

Abstract: The memory access patterns of motion estimation algorithms were analyzed, the cache efficiency, related issues and the causes were revealed when they run on general PCs and digitial signal processors. A new cache efficient video data layout method, i.e., overlapped tiled layout method, was presented. The results show that this new data layout method can be used to resolve the cache split loading issue and greatly reduce the cache miss rate. The new layout method can speed up the spiral full search motion estimation process for 28% without change of the motion estimation algorithm itself. So, the use of the overlapped tiled layout method can speeds up the motion estimation process.

Key words: video data; memory; access efficiency; motion estimation

运动估计(Motion estimation，ME)是视频处理系统中的一个重要组成部分。为提高运动估计的效率，一方面，人们致力于提高算法本身效率的研究，提出了许多精巧的运动估计算法；另一方面，研究了运动估计算法的内存访问模式对运动估计效率的影响。研究结果表明，运动估计算法的内存访问模式对运动估计效率有至关重要的影响。在运动搜索引擎的硬件实现中，Chen 等[1−2]研究了如何重复使用已经被调入到引擎的数据从而减小数据总线的压力，这一类运动估计算法的内存访问模式通常是规则而又连续的，运动估计算法的控制简单。目前，在软件实现的运动估计算法中，几乎所有的研究都集中在如何有效地减少搜索点数而又保证搜索精度等方面，这一类算法内存访问模式通常是不规则的，跳跃的，控制逻辑复杂，典型的有三步搜索[3]、菱形搜索[4]、PMVFAST[5]等等，而对软件实现的运动估计算法内存访问效率的研究较

收稿日期：2007−01−10
基金项目：湖南省教育厅基金资助项目(04C604)；邵阳学院重点科研基金资助项目(2003B08)
作者简介：赵乘麟(1965−)，男，湖南邵东人，副教授，硕士，从事图形图像和视频技术的研究
通讯作者：赵乘麟，男，副教授；电话：13087399858；E-mail: chenglinzhao@126.com

少。Alex 等[6]从运动估计的内存访问效率上对运动估计算法的实现进行了一些讨论，提出了使用一个软拷贝缓冲器存储搜索区，从而避免跨线访问数据的问题，但随之产生的问题是：如果搜索区超出[−32，32]，由于拷贝数据的过程很昂贵，而且拷贝数据并不能降低高速缓存跑靶率，在这种情况下，该算法的有效性就会显著下降。Yang[7]则根据运动估计算法的内存访问量分析了一些运动估计算法的功耗和效率，但并未提出任何解决方法。Kulkarni 等[8]提出在编译器的层面来优化媒体数据的存储方式，但其提出的解决办法在实际中很难实现，即使实现也会由于程序变得复杂，其最终效果难以保证。以上所有的研究除文献[8]外全部假定原始视频数据是按常用的平面存储方式存放，即图像总是从左至右，从上到下逐行逐点存放到连续内存里面，有时为了使运动估计方便也会在图像边界的最左边和最右边填充一些冗余点，以避免进行越界检查，但总体存储方式不变。这种平面存储方式不适合用于运动估计，是直接造成运动估计算法的高速缓存效率不高的原因，降低了一些设计精巧的运动估计算法的效率。

1　运动估计算法的内存访问模式

1.1　高速缓存结构

通用计算机或数字信号处理机通常都采用分集关联高速缓存作为其一级缓存，如 Pentium 4 Northwood 机型就采用了如图 1 所示的 8 KB 四路分集关联高速缓存作为其一级缓存，其每一线的长度是 64 字节，8 KB 空间共有 128 线，同一行中的四线属于同一集，纵向的 32 线属于同一路， 每一线对应的还有 1 个标签用来快速查寻在缓存中的 64 字节长的内存区，这种缓存结构具有以下特点：**a.** 内存在缓存中的调入和调出以线长 N 为单位，如图 1 中 N 为 64 字节；**b.** 以 N 字节对齐的内存地址及其后连续的 N 字节的内存位于同一线中；**c.** 当新的数据要求被调入时，被调出的是将要被放入的那一集中的最近、最少被用的那一线。

影响一级缓存效率的情况有以下几种：**a.** 数据未命中缓存(cache miss)；**b.** 缓存线冲突(cache conflict)；**c.** 跨线数据访问，即要访问的数据分属于 2 个相邻缓存线(cache split load)。

这 3 个因素对程序的效率都有很大的影响。在此，以 Intel Pentium 4 处理器为例，对比分析数据未命中缓存和命中的数据访问时延。当访问一个 4 字节整数时，若被访问的数据恰好在一级缓存中，则命中，其时延是 2 个时钟周期；若不命中，则时延至少有 14 个时钟周期；而跨线数据访问就相当于倍增了数据访问量，既占用了更多的缓存线，又增大了数据未命中缓存的概率。

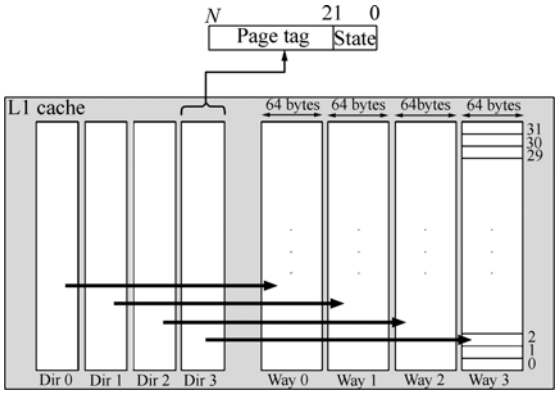

图 1　四路分集关联高速缓存结构

Fig.1　Structure of 4-way set associative cache

运动估计的过程就是在给定的参考帧内为当前块找一个最优的匹配块的过程。一个常用的匹配准则可以用以下公式表示：

$$S(X)=\sum_{i=0}^{N}\sum_{j=0}^{M}|x_{ij}-y_{ij}|。\tag{1}$$

其中：X 为当前块；x_{ij} 为当前块的第 i 行、第 j 列的亮度; y_{ij} 为参考块第 i 行、第 j 列的亮度。为简化分析，暂时假设 N 和 M 都为 16，且只搜索整点象素运动矢量。

1.2　运动估计算法的数据访问模式

以全搜索算法为例分析运动估计算法的数据访问模式。全搜索算法是在给定的搜索区内逐个匹配所有块，其中 S 最小的块和当前块的位移就是最后的运动矢量。通常搜索区是在参考帧内以当前块同位置的块的最左上角点为中心的一个矩形区。例如，如图 2 所示的运动搜索示意图中的阴影区就是宏块 A 的搜索区。

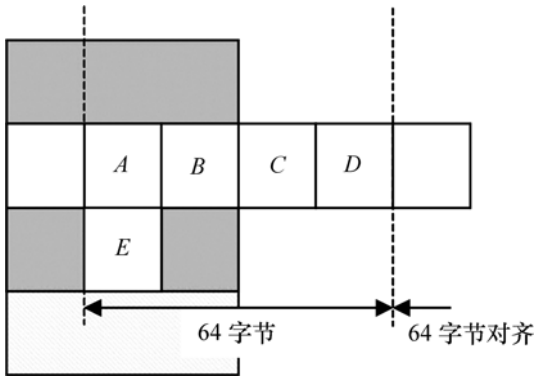

图 2　运动搜索示意图

Fig.2　Diagram of motion estimation search

在平面存储方式下，当前块的最左上角的点总是16 字节对齐的，因此，同一行的点都会在同一缓存线内，当处理器访问当前块时，不存在跨线访问问题，而参考块则完全不同。从图 2 可以看出，假如运动搜索范围为[−16，16]，且宏块 *A* 的起始位置是 64 字节对齐，则宏块 *A* 左边所有的搜索点，即宏块 *A* 的运动矢量的 *X* 分量为负的所有点都会跨缓存线访问数据。同样，宏块 *D* 右边的点也存在跨线访问数据的问题，而宏块 *B* 和 *C* 则不会。由此可见，有一半宏块的运动搜索过程会存在跨线访问数据的问题。

利用图 2 还可以分析平面存储的另一个问题，即当处理宏块 *E* 时，它的搜索区即条纹阴影区，与宏块 *A* 的搜索区重复的那些数据基本上被置换出一级缓存(对于 352×288 以上的图像成立)，因此，不可避免地要重新发生一些数据不命中缓存的情况。

以典型的 D1 精度的图像为例，为方便分析，把图像的宽度从 720 扩充到 768 点宽，以确保每行 64 字节对齐。这样，每行要占用 12 个缓存线，由于搜索区的垂直高度是 48 行，故完成一行宏块的运动估计共需调入数据 12×48=576 线，远大于 128 线的缓存容量。由图 2 可知，宏块 *A* 由于在处理其左邻宏块时已经完成数据的调入(数据调入以缓存线为单位)，故不发生数据不命中事件，同理，*B* 和 *C* 亦都不发生，*D* 则要调入相邻的右边和右上以及右下宏块所在的 48 线数据，故发生 48 次数据不命中缓存的事件。所以，平均每个宏块产生 12 次数据不命中事件，每个宏块的搜索区平均占用的缓存线数为 12。

上面的分析忽略了缓存线冲突，缓存线冲突的原因是缓存是按线分集组织的，内存区不可以随意放进任何缓存线中，而是根据其地址不同放进不同的集合中，可以用文献[6]中的公式进行计算。这样，平面式存储运动估计的缓存效率会进一步降低。

高度优化的 Intel H264 编码器(原始数据在编码中全部用平面存储方式存放)在支持 Hypter Thread 的 Pentinum 4 Prescott 3.2 G 机器上运行时使用 Vtune 6.0 实时提取的有关高速缓存效率的抽样数据如表 1 所示。Pentinum 4 Prescott 机型有 16 KB 8-路一级数据高速缓存，1 MB 8-路二级通用缓存。在编码过程中都使用了 3 个参考帧，运动搜索范围如下：*X* 方向为(−23，23)，*Y* 方向为(−15，15)，实验中每次编码 300 帧。

如果这一指标大于 2(经验数据)，通常意味着数据未命中缓存对程序有很大的负面影响，可以看出：即使是高度优化的 H.264 编码器，数据的缓存跑靶率也是一个严重的问题。因此，常用视频数据的平面存储方式直接决定了运动估计的内存访问模式。运动估计过程在平面存储方式下存在高速缓存效率不高和跨缓存线访问数据的问题。

表 1 Intel H264 编码器高速缓存效率

Table 1 Intel H264 encoder's Cache performance efficiency

	352×288 CIF sequence	720×480 D1 sequence
Clock ticks (×3，200，000)	7，229	20，531
Load retired(×234，939 sample)	28，623	81，254
1st level cache load miss retired (×5，581 sample)	20，233	65，673
1st level cache load hit rate	98.321	98.080
1st level cache load miss performance impact	4.881	5.579
Split load retired(×536 sample)	21，995	56，206
Split load performance impact	1.529	1.376

注：1st level cache load miss performance impact 定义如下:

1st level cache load miss performance impact = ((1st level cache load miss retired * 22)/clockTicks)*100。

2 视频数据的叠瓦式存储

从上面的分析可以看出，由于数据是按行存放，造成当前要使用的数据在空间上不连续。但改成把数据按宏块组织即同一宏块的数据连续存放，若要访问的数据分别来自 4 个宏块，则程序本身变得复杂，不利于编译器对程序进行优化。实验结果表明：用这种数据组织方式进行运动估计时，因为缓存效率的提高所节约的时间还不能抵消程序复杂所带来的负面效果。基于以上原因，提出了叠瓦式数据组织方式，即：从上到下为相邻每 2 行的宏块对，如图 3 所示，自左至右垂直扫描其搜索区直到最后 1 列。很明显，宏块行对(2*n*，2*n*+1)的搜索区和宏块行对(2*n*−2，2*n*−1)的搜索区，以及(2*n*+2，2*n*+3)的搜索区都会有部分重叠。如图 4 所示，从左到右为相邻每 2 列的宏块对，自上到下水平扫描其搜索区直到最后一行。一个显而易见的事实是，当用这种存储方式组织数据时，存储空间大约是原来的 2 倍，除最左边(或最上边)的 2 列(行)宏块和最右边(或最下边)的 2 列(行)宏块外，所有宏块在同一帧内都被存储 2 次。直觉上会以为数据访问量将是原来的 2 倍，但由于叠瓦式存储能避免重复调入前面已经访问的数据(图 2 中宏块 *A* 和 *E* 的搜索区重叠的部分将只被调入 1 次)，实际数据访问量反而下降。

仍以前面的例子对比分析当图像按水平叠瓦式存

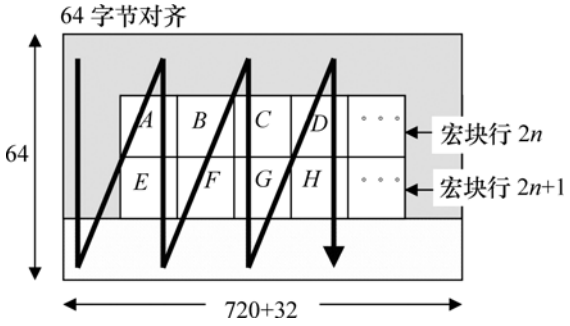

图 3　水平叠瓦式存储示意图

Fig.3　Diagram of horizontal overlapped tiled format

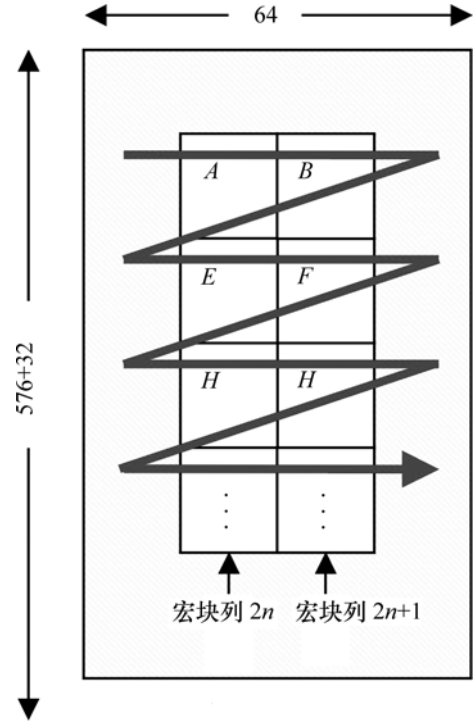

图 4　垂直叠瓦式存储示意图

Fig.4　Diagram of vertical overlapped tiled format

储时运动估计的缓存效率，并假设搜索区是[−16，16]，可得出以下结论：

a. 叠瓦式存储没有跨线访问数据的问题。这是由于属于同一列的数据属于同一缓存线，所以，无论是水平还是垂直移动搜索点，数据都属于同一线。如果搜索区大小超出[−16，16]，则不使用当前的“瓦”，而使用紧邻的另一块“瓦”来计算 *S*。其原因是每一个宏块都有 2 个拷贝，分属于不同的“瓦”，如果 1 个宏块在当前“瓦”内是个边缘块，只有部分落入当前“瓦”，那么，它在其紧邻的“瓦”中一定是个中心块，完全落入其中。

b. 叠瓦式存储数据未命中缓存的次数只有平面存储的 70%左右。以图 3 为例，仍然假设图像是典型的 D1 精度图像，那么水平块“瓦”的宽度为(720+32)，高度为 64， 运动估计时一次处理 1 对数据，共调入(720+32)线进入高速缓存，不需重复调入任何数据。故平均每个宏块产生(720+32)/90=8.36 次数据未命中缓存事件，只有平面存储的 70%左右。

c. 叠瓦式存储不存在缓存冲突问题。这是由于同一宏块的数据在存储空间上也相邻。

d. 叠瓦式存储所要求的存储空间大约是平面存储空间的 2 倍。

3　叠瓦式存储的实现

叠瓦式存储在具体实现时会遇到如下几个问题：

a. 若要求显示重建的图像，则需要做格式转换，即从叠瓦式存储转换到平面存储。这个问题容易得到解决。因为编码器通常并不需要显示重建图像。另一方面，对编码器来说，格式转换的开销相对高速缓存效率变高带来的优点来说可以忽略不计。在本实验中，在每一帧的结尾都进行了 1 次格式转换。结果表明，叠瓦式存储仍然明显优于平面存储。

b. 在实际的视频压缩标准中，对宏块的编码顺序通常都是遵循平面存储的顺序，而且在真正进行运动矢量的预测编码时，都要从其相邻的宏块预测当前宏块的运动矢量，然后，预测编码。这个问题对于 H.264 来说很容易解决，因为 H.264 支持 MBAFF 选项，当 MBAFF 选项打开时，宏块编码顺序正好是水平叠瓦式存储格式的存储顺序，即 1 次处理 1 对垂直相邻的宏块，因此，不需要做任何设计上的改变。而对于 H.263 或 MPEG-4(Part 2)，需要水平叠瓦式存储的宏块处理顺序按图 5 所示顺序改变。其中宏块上面的数字就是其编码次序，这种改变对高速缓存效率并无影响，因为处理 2，4，6 和 8 等偶数宏块时，它们

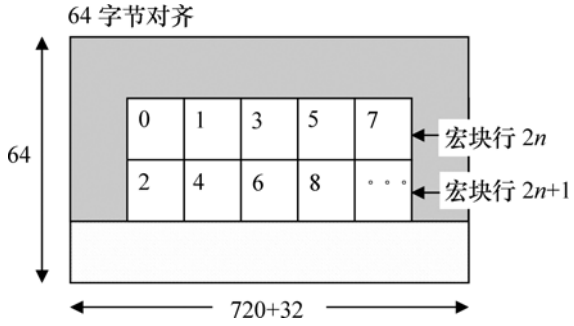

图 5　宏块处理顺序示意图

Fig.5　Diagram of macroblock processing order

所需要的数据仍然在高速缓存中，并不会因为处理其左上宏块时多调入 16 线数据而把它们的数据置换出高速缓存。事实上，如果编码器出于某种特殊的考虑，需要用更多的相邻宏块信息以帮助找到最佳预测运动矢量或量化参数等，可以使用图 6 所示的更一般处理顺序。当然，若 m 等于 1 行内的宏块数，则水平叠瓦就退化成平面存储格式。M 若取值过大，则高速缓存效率将显著降低。

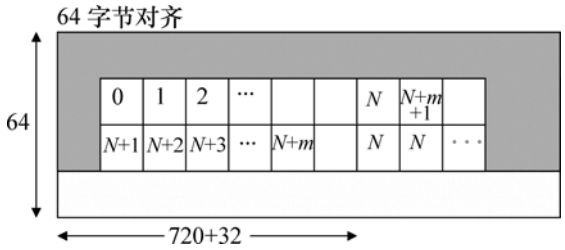

图 6 通用宏块处理顺序示意图

Fig. 6 Diagram of general macroblock processing order

4 实验结果

实验中，分别按水平叠瓦式存储(m=1)和平面存储的同一图像序列中的每一宏块在[−16，16]区间进行螺旋全搜索运动估计[10]，并为水平叠瓦式存储的每一帧进行 1 次格式转换。在寻找最优匹配块时使用 PDE 排除法，运动估计算法模块使用 MMX 指令实现。表 2 所示为 2 种存储方式的高速缓存效率的抽样数据。实验在 1 台 3.2 G Pentium 4 Prescott 机上完成，图像尺寸是 4CIF(704×576)，分别完成 26 帧运动估计(只做 26 帧的运动估计是为了加速数据的采集过程，参考帧数为 4)。由表 2 可见，这 2 种存储方式下所需的时钟周期不同，两者比值为 163 68/227 32=0.72，因此，水平叠瓦存储方式使全搜索运动估计速度提高 28%。

表 2 平面存储和水平叠瓦方式的比较

Table 2 Comparison between two layout methods

缓存效率采样数据	平面存储	水平叠瓦
Clock ticks	22, 732, 800, 000	16, 368, 000, 000
Load retired	9, 539, 987, 757	8, 476, 991, 915
1st level cache load miss retired	14, 371, 104	11, 989, 056
1st level cache load hit rate/%	99.849	99.859
1st level cache load miss performance impact	0.632	0.732
Split load retired	14, 742, 530	9, 691
Split load performance impact	1.946	0.002
2nd level cache load miss retired	600, 340	462, 265

5 结 论

a. 采用叠瓦式存储方式完全根除了运动估计过程中出现的数据访问跨缓存线访问的问题。

b. 使用叠瓦式存储方式，一级和二级缓存跑靶次数减少。

c. 叠瓦式存储能显著地加速全搜索算法运动估计过程；从原理上推断，这种新的存储方式对别的运动估计算法和运动补偿过程同样可以起到加速的效果。

参考文献：

[1] Chen C Y, Huang C T, Chen Y H, et al. Level C+ data reuse scheme for motion estimation with corresponding coding orders[J]. IEEE Trans on CSVT, 2006, 16(4): 553−558.

[2] Tuan J C, Chang T S, Jen C W. On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture[J]. IEEE Trans on CSVT, 2002, 12(1): 61−72.

[3] Li R, Zeng B, Liou M L. A new three-step search algorithm for block motion estimation[J]. IEEE Trans Circuits Syst Video Technol, 1994, 4(4): 438−442.

[4] Zhu S, Ma K K. A new diamond search algorithm for fast block-matching motionestimation[J]. IEEE Transactions on Image Processing, 2000, 9(2): 287−290.

[5] Tourapis A M, Au O C, Liou M L. Predictive motion vector field adaptive search technique (PMVFAST) enhancing block based motion estimation[C]//Proc SPIE Conf Visual Commun Image Process. Japan, 2001: 883−892.

[6] Lopez-Estrada A A. Understanding memory access characteristics of motion estimation algorithms[EB/OL]. [2007−04]. http://www.intel.com/cd/ids/developer/asmo-na/eng/182345.htm.

[7] YANG Sheng-qi, Wolf W, Vijaykrishnan N. Power and performance analysis of motion estimation based on hardware and software realizations[J]. IEEE Transactions on Computers, 2005, 54(6): 714−726.

[8] Kulkarni C, Ghez C, Miranda M, et al. Cache conscious data layout organization for conflict miss reduction in embedded multimedia applications[J]. IEEE Transactions on Computers, 2005, 54(1): 76−81.

[9] Hayes B. Differences in optimizing for the pentium 4 processor vs the pentium III processor[EB/OL]. [2007−04]. http://www.intel.com/ cd/ids/developer/asmo-na/eng/44010.htm.

[10] JVT Reference Software Ver11.0[EB/OL]. [2007−04]. http://iphome.hhi.de/ suehring/tml/.