第 4 卷(A 版) 第 10 期
1999 年 10 月

中国图象图形学报
Journal of Image and Graphics

Vol. 4(A), No. 10
Oct. 1999

运动估值中的最优程序技术

刘德春
(南阳理工学院电信工程系，南阳 473009)

周孝宽
(北京航空航天大学图象中心，北京 100083)

摘 要 由于运动估值算法的特殊性，程序实现技术对匹配速度有很大影响。通过对基于块匹配法的全匹配算法的具体分析和对一般 C/C++语言实现方法的改进，给出了在运动估值中可以普遍采用的最优程序技术。文中给出了全匹配算法和典型快速算法的速度与精度以及改进前后的对比实验结果。

关键词 运动估值 软件技术

0 引 言

运动估值是序列图象压缩领域运动补偿技术中的一个重要环节，其效率和精度直接影响图象的压缩效果。众所周知，除了全匹配搜索算法(或穷尽搜索法 FSM)外，还有其它各种不同的快速算法，如对数法、对偶法、三步搜索法[1]等。一般的快速算法在速度上要比全匹配搜索法快一个数量级，但其估值质量要比后者低得多。就目前来说，各种快速算法并没有得到广泛认可，全匹配算法在不同场合也有应用。显然，图象的恢复质量在图象压缩中应是第一重要的。由于全匹配算法在估值算法中准确度最高，随着机器硬件速度的不断提高，其应用会越来越受到重视。如文献[2]在可视电话编码压缩中就提出了不同的全匹配算法(简记为“筛选法”)，文献[3]援引并介绍的“生长法”也对全匹配算法进行了搜索方向的改进并加了阈值限制等。实践中发现，同一种估值算法，由于实现的手段不同，速度的差别可以很大，就全匹配算法来说，最高速度和最低速度可以相差几十倍，说明有很大潜力可挖。在算法实现上除了编程技术各有差别外，还有重要一点，就是用常规的 C 语言编程方法实现的运动估值算法存在着许多影响运算效能的操作，特别是在指令级上的影响因素常常被语言本身的习惯用法所掩盖，不被人重视。但程序实现的优劣对运算速度的影响程度甚至胜过某些改进算法本身。因此，充分利用运动估值中数值运算的特点，把匹配算法中的程序实现技术用到最优，对于运动估值算法的软件实现有着重要的意义。

收稿日期：1998-12-16；收到修改稿日期：1999-02-02

1 块匹配法和运动估值的特点

在运动图象编码中二维运动估值被广泛应用于运动补偿帧间预测、帧间内插中。帧间运动可以近似为本帧(第 k 帧)内一个区域或多个区域相对于参考帧(如第 $k-1$ 帧)的逐段平移。块匹配法是一种基于模式匹配的位移估值法。其基本思想可以描述如下(如图 1)：首先，本帧(第 k 帧)图象被均匀划分成 $M\times N$ 象素的矩形块(称为子块)，并假定位于同一子块内的所有象素具有同样的位移，则对每一子块只需要计算一个位移矢量。每一子块在参考帧内预先确定的一个搜索窗内的所有位置上进行位移匹配(互相关)运算。这里搜索窗大小为 $(M+2h)\times(N+2v)$，其中 v,h 分别为水平和垂直方向的最大位移(整象素数)。对每一位移 $[(i,j): -h<i<h, -v<j<v]$，在给定的匹配准则下，计算出相应的代价(失配函数值) $D(i,j)$，而相应于使 $D(i,j)$ 为最小的 (i,j) 给出了该块的位移矢量。

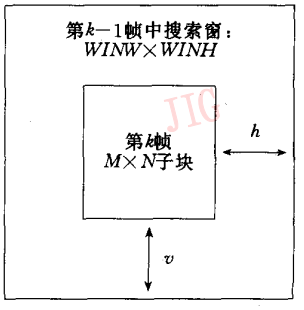

图1 块匹配法示意

目前常用的匹配准则有均方误差(MSE)和绝

对值误差(SAD)准则等。其表示形式为：

$$\mathrm{MSE}(i,j)=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}|f_k(m,n)-f_{k-1}(m+i,n+j)|^2 \quad (1)$$

$$\mathrm{SAD}(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}|f_k(m,n)-f_{k-1}(m+i,n+j)|^2 \quad (2)$$

其中，f_k 表示第 k 帧灰度值，i,j 分别为水平、垂直方向的偏移量。文中算法采用 SAD 准则。

显然最优匹配应是全匹配搜索法，即对窗内的每一点都计算 SAD 值，最小的 SAD(i,j)值即对应于最优匹配，每个搜索窗共需计算$(2h+1)\times(2v+1)$个 SAD 值，其运算量是巨大的，这也正是运动估值的特点。设图象的高度为 $HIGHT$，宽度为 $WIDTH$，搜索窗的宽高分别为 $WINW$ 和 $WINH$，宏块大小为 16×16，那么对一幅图象进行估值时，仅求 SAD 累加和的加法(含减法)运算就有 $HEIGHT/16\times WIDTH/16\times(WM-15)\times(WN-15)\times16\times16\times2$ 次，若以 352×288 图象、32×32 搜索窗为例，要执行加法 5800 多万次；仅估计出一个宏块的运动矢量就要执行加法近 15 万次。这其中还不包括求最小值时所进行的比较运算，事实上它也要作减法运算，另外还有求绝对值等运算未予统计。这意味着，估值运算当中的任何冗余操作都会给运算时间带来很大的开销；相反，去除这些冗余操作，即便是很少的改善，也可能使匹配运算速度得到较大提高。匹配算法中的核心运算是 SAD 值的计算，而影响程序执行效率的冗余操作也主要体现在这里。下面就针对 SAD 运算的求值问题给出具体改进措施。

2 选择退出宏块之 SAD 运算的最佳判断点

既然估值时要在搜索窗内匹配最小值，人们自然会想到，应在已搜索过的搜索块中记下目前最小的 SAD 值 MIN_SAD，如果下一个正在搜索的候选匹配块的 SAD 运算值已经超出 MIN_SAD 时，就应该停止该块剩余象素的匹配运算(放弃该块)，以节省时间。文献[3]推荐的“生长法”用到了此方法。但在该方法中提出，每作一次加法运算后都要判断 SAD 的累加值是否超出 MIN_SAD，是不完善的。原因是它忽视了判断语句占用的时间。判断语句不仅占用判断指令，且要占用额外的减法运算时间。这样做的结果额外增加了和象素数数目同样多的判断语句，对于那些最终未能退出运算的块，实质上又付出了较多的代价，如对 16×16 的一个宏块，就至少增加了 512(16×16×2)条指令的执行；即使对于满足退出计算条件的宏块，这样做也是不经济的。最佳的选择是把判断放在宏块的每一行象素运算之后。通常宏块的每一行构成循环的一个自然单位，一行内的数据由于地址的连续递增，数学运算的速度可以比较快；且以一行数据作为计算单位时，执行判断语句少，在每一个判断指令执行时，当前块的运算结果超出 MIN_SAD 的概率远远超出前者，付出的总代价小。试比较两种判断方式，当本块不满足退出条件时，前者要浪费 256 个判断语句，而后者为 16 个判断语句，后者大大优于前者；当满足条件时，前者多用的判断语句个数等于计算过的象素点数(通常均远大于 16)，而后者只等于计算过的行数(≤16)，另外只比前者多出了不超过 16 个象素点的 SAD 值累加运算，整体效果一般也优于前者。全匹配的实验结果表明，后者与前者相比，平均匹配时间快 17.3%以上。无论如何，由于判断语句的加入，排除了许多非匹配块的无为运算，所以引入它肯定比没有要好。此方法既适用于“生长法”，也适用于传统顺序全匹配或其它算法，只是在用于“生长法”时效果更加突出。

3 用汇编语言除去块的移动和计算 SAD 时的全部冗余操作

估值运算过程占时最多的操作主要集中在两个方面：每个定义的搜索窗内的宏块位置的控制移动以及宏块 SAD 值的运算。这两个方面的程序实现都存在象素的地址指向问题，在用 C/C++语言编程实现时存在较多的冗余操作。首先，最简单直观的实现方法是用两维数组，很容易就可实现位置指向的移动和 SAD 值的计算。但事实上用数组进行数据操作时，在机器内部要先做地址的计算转化工作，特别对于二维以上的数组，还要根据数组的宽度算出当前的地址，浪费的机时更多。如前所述，由于估值时匹配运算次数较多，使得数组元素频繁操作，大大降低了运算效率。再者是用指针指示搜索窗和宏块的地址，如图 2 所示。其中 pwstart、pw、pc 分别指向搜索窗、候选块、待匹块的首地址。通常一帧图象的数据量比较大，在读进内存时普遍要超出 64K 范围，即数据范围跨段，在使用 C/C++编程时(大模式)须用巨指针

或有条件地使用长指针指向数据元素。使用指针是C语言加快数据处理的最好途径,无疑将大大提高程序的执行速度。如用 pw_temp 和 pc_temp 分别代表 pw 和 pc 所指宏块内的行内首地址,则计算每块 SAD 值的循环体运算的 C 语言算法是:

```
sad←sad +| (*pw_temp++)−
         (*pc_temp++)|                    (3)
```

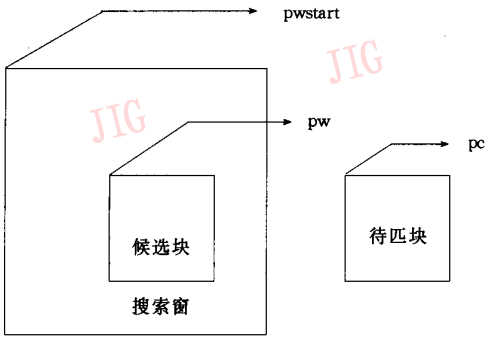

图 2　指针指向示意

巨指针和长指针都是 32 位指针,如果把巨指针规格化后赋予长指针,就可以使长指针在不超过 64K 的范围内代替巨指针进行正向(非逆向)的变址操作,而搜索窗的范围通常是符合条件的。这样做的好处在于,指针在变址操作时缺省了其高 16 位地址的运算。实践的结果表明,使用巨指针后比用数组速度提高 40.25%,结合在局部范围使用长指针,比单纯使用巨指针速度又可提高 73.7%,效果是很明显的。

尽管上述做法已经不错,但分析发现,程序对运动估值的特定算法来说,依然还存在着许多不必要的操作:

(1) 巨指针在每次变址(增或减)时都要进行规格化运算处理。

(2) 通过指针向内存读取每个象素点的数据时,每次都需向数据段寄存器 DS 进行传送操作,以免发生段值改变的错误。

(3) 在(3)式中每做一次减法运算都要调用一次求绝对值的函数,它必将伴随频繁的出入堆栈操作。

虽然上述的每项各占用时间很少,编程时通常根本不予考虑,但对于估值运算的超大运算量来说,所占总时间也很可观,不容忽视。根据(3)式的运算,采用 C 与汇编语言混合编程的方法,就能够从根本上去除存在的弊端。扼要说明如下:

直接用汇编指令实现(3)式的计算,避开用 C 的指针操作数据。虽然进入内存的每帧图象数据可能处于不同的数据段之间,但可在预测帧内每次搜索窗定义之后,把搜索窗的起始地址进行规格化处理,赋予长指针,这样从该指针看,搜索窗内的数据就能保证全部在同一个数据段内;之后把指针的段值部分赋予数据段寄存器 DS。被预测帧内的宏块也做类似的处理,把段值部分赋予附加段寄存器 ES。在两帧中的宏块匹配之前,把起始地址偏移值(指针的低 16 位)各赋予不同的变址寄存器(如 SI、DI),以后执行带段超越的加减法操作。由于在整个搜索窗的块匹配运算期间段寄存器只装载一次,所以数据的内存读取时间就降到了极限,速度必将得到很大提高(要求 SI 或 DI 进行正向寻址)。另外,对于图象数值的 8 位无符号数(单波段象素灰度值),其差值的绝对值运算,可以避开调用函数操作,只需用汇编指令作一次减法,再加上溢出判断就可实现,这比求普通数值的绝对值要容易。其依据是:对无符号数 X、Y,设 $A=X-Y$,当无溢出时,进位标志 $Cy=0$,则 $X\geq Y$,其差的绝对值为:$X-Y=A$;有溢出时,$Cy=1$,则 $X\leq Y$,此时绝对值应为:$Y-X=2^8+Y-X=2^8-(X-Y)=2^8-A$,即 A 的补码($\mathrm{mod}=2^8$)。也就是说,当无溢出时累加器的结果就是绝对值本身(无符号数);而当溢出时,对该溢出的结果直接求补即为正确的绝对值(无符号数),无需再重作减法,而求补运算比减法运算的机器数要省得多。经过以上两方面改进后,既大幅度地减少了数据的内存读取时间,又使核心运算(3)式的实际执行指令数最少,实践中匹配速度比应用 C 语言指针的最快情况提高 71.2%。需要说明的是,上述改进是提高核心运算效能的必要处理,并非追求对整个程序的汇编书写,在改善程序执行的同时,不影响其可读性和易维护性。

4　结果与结论

从 Susie 标准序列图象中取 11 幅 352×240 大小的图象,以第一帧为基础帧做正向预测,宏块大小为 16×16,搜索窗口的大小为 32×32,在 pentium 100 微机上执行。为比较匹配算法运动估值的准确性,规定当前被预测帧的宏块均用最小误差匹配宏块去替换。表 1 列出了传统顺序全匹配算法采用不同实现方法包括文中改进技术后的执行情况(顺序全匹配系指从左到右,从上到下的搜索)。表 2 给出了各种全匹配算法和典型快速算法对偶法的匹配结果,以及加入本文第 2、3 节所述改进技术后的对比结果。其中为方便比较,"生长法"去掉原文的加阈值

处理,成为真正的全匹配;"筛选法"的 $M(x,y)$(搜索窗中每个宏块所对应的象素和)在匹配之前用快速算法计算一次,且时间不计入总估值时间。

表 1 顺序全匹配不同实现的结果

指 标	实现方法				
	数组	巨指针	长指针	用文中第 2 节方法改进	用文中第 3 节方法改进
平均每帧匹配时间(s)	40.11	26.98	7.0	3.83	1.80
比前项降低(%)		40.25	73.7	45.28	53

表 2 几种不同估值方法的结果比较

指 标	估值算法			
	顺序全匹配	生长法全匹配	筛选法全匹配	对偶快速算法
原方法平均每帧估值时间(s)	7.0	3.56	5.15	0.23
改进后平均每帧估值时间(s)	1.80	1.10	2.88	0.16
恢复帧相对于原帧信噪比(dB)	35.885	35.885	35.885	33.193

从以上的匹配结果可以看到,文中改进技术对于各种估值算法的匹配速度均有明显的改善。尤其加入改进技术后的生长法在已知的全匹配算法中是速度最快的算法。本文论及的程序改进技术是估值算法在实现上的优化,不仅适用于全匹配算法,也适用于各种快速算法。

参 考 文 献

1 林富宗,陆 达. 多媒体与 CD-ROM. 见:运动补偿,清华大学出版社,1995.3,354~359.

2 王辉柏,张春田. 极低码率视频编码中块自适应分割运动估值. 中国图象图形学报,1998,3(6):466~470.

3 何圣静. 多媒体技术及其应用. 北京:北京理工大学出版社,1996,44~46.

4 Wei Li. Successive elimination algorithm for motion estimation. IEEE Transactions on Image Prosessing, January 1995, 4(1): 105~107.

5 刘德春. 动图象压缩方法研究[学位论文]. 北京:北京航空航天大学图象中心,1998.

刘德春 1998 年毕业于北京航空航天大学,获工学硕士学位,现为南阳理工学院讲师,电信工程系副主任。主要从事信号处理,图象压缩编码,多媒体技术等领域的教学和研究工作。

周孝宽 北京航空航天大学教授,博士生导师,北京宇航学会和中国图象图形学会常务理事,中国用户协会图象分会理事。主要研究方向为图象压缩编码,分形图象学和医用图象的识别与处理等。

Best Programme Technique in Motion Estimation

Liu Dechun

(*Department of Electronics and Information Engineering, Nanyang Institute of Science and Engineering, Nanyang* 473000)

Zhou Xiaokuan

(*Image Center, Beijing University of Aeronautics and Astronautics, Beijing* 100083)

Abstract Due to the inherent characters of motion estimation algorithms, the matching speed is greatly influenced by the implementation technique. Based on the detailed analysis of the block based exhaustive searching algorithm and making full use of the programming language C/C++, a best program technique suitable for motion estimation is given. The results of using the proposed technique for different motion estimation algorithms are also given.

Keywords Motion estimation, Software technique