运动估值的快速算法及其实现

刘德春 周孝宽
(北京航空航天大学 宇航学院)

摘 要 现有的许多有关运动估值的快速算法,都存在着匹配速度快与匹配精度差的矛盾.文章在分析已有典型快速算法优缺点的基础上,提出了解决这一矛盾的分步逼近的新算法——迂回逼近法.算法选择了快捷和更为准确的搜索路径,且对程序的实现技术作了有效改进,其最终匹配结果具有全匹配算法的精度和典型快速算法的速度.文中说明了算法原理、程序技术和对比实验结果.

关键词 图像处理; 图像编码; 匹配; 运动估值
分类号 TN 919.8

1 问题的提出

为提高运动图像的压缩倍数, H.261, H.263, MPEG1, MPEG2 等均把运动估值作为一项关键技术使用.由于运动估值在用全匹配方法实现时运算量巨大,耗费时间长,难以实现运动图像的实时压缩,所以人们陆续提出了各种不同的快速算法,如:三步法、对数法、对偶法[1]等.这些快速算法的突出优点是运算速度一般比全匹配搜索算法约快一个数量级,明显的缺点是匹配精度较差.以快速算法中的对偶法为例,该方法与全匹配算法的匹配精度作比较,取 11 幅标准图像(352 × 240 单波段),以第 0 帧作为基础预测帧(16× 16 宏块, 32× 32 搜索窗),其余帧均用匹配到的最小失真块去替换当前块,则用以上 2 种方法所得的估值图像相对于原图像的平均信噪比分别为 33.193 dB 和 35.885 dB,相差 2 个多 dB.上述各种快速算法的速度之所以快,主要是因为大量减少了搜索窗内的搜索点数,而精度差则是因为受其指定的搜索路径的范围限制,不能保证绝大多数宏块都找到最佳匹配位置.理想的快速算法应该做到,在保持速度优势的前提下以最小的精度损失去逼近全匹配的最优估计.本文在发掘现有快速算法"对偶法"优点的基础上,通过选择更为准确的搜索路径,提出了实现上述要求的快速算法——迂回逼近法.

实用的快速估值方法应该不仅要有优质的算法,还应该有高效的程序实现技术,为此本文在提出新算法的同时,又从减少算术运算的总次数和简化核心运算的指令等方面入手,使算法在计算机上得以更快地实现,迂回逼近法的最终结果基本上达到了全匹配的精度和典型快速算法的速度.

2 对偶算法

对偶算法是一种有代表性的快速算法,其匹配准则采用平均绝对值差准则,在下面的叙述和匹配运算中采用它的简单形式——绝对值差准则(SAD).

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

其中 f_k 表示第 k 帧灰度值; i,j 分别为搜索窗内水平、垂直方向的偏移量; $M \times N$ 为图像子块的大小.

对偶算法的主要思想是,在搜索区域(窗)中某一起始点(如预测帧的对应位置块)开始,先在横向(X 向)上确定最小失真的方向,继之朝这方向进行连续的比较匹配,找到匹配最小失真点(位置);然后再以此点为始,在纵向(Y 向)确定最小失真方向,接着执行与 X 向同样的搜索过程,该过程结束时所找到最小失真点即被认为是最终估值点.设若用全匹配方法需要搜索 $M_1 \times N_1$ 个点的话,则用对偶法时至多搜索 $M_1 + N_1$ 个点,所以搜索速度要快得多.对偶法的优点还表现在路径直接,易于实现,大致反映了匹配数据的分布规律,但是由于其搜索的步数过于有限(只有 2 步),在该有限的路径上相当一部分块不能找到最佳匹

收稿日期: 1998-03-24 第一作者 男 33 岁 讲师 100083 北京

配位置,所以它的匹配精度与全匹配算法相比精度损失较大.

3 迂回逼近法

本算法包括优化搜索路径和合理选择搜索起始点2个部分.

3.1 分步逼近的搜索路径

经过对典型动图序列图像的全匹配运动估值的误差散布进行分析和统计发现,搜索窗中各个搜索块的误差散布基本都是单峰值,如图1所示(数据图为32×32搜索窗内的不同搜索位置上的SAD值分布的部分截取数据).此单峰值即为最小误差点,而在此点的附近,数据的分布没有一个可以准确描述的规律.对偶快速算法搜索到的误差匹配点(块)只适用于一部分情况,还有相当一部分估值点距离最小估值点仍较远(如图1中,从SAD值678处始,用对偶法可得估值点SAD值为473,而最小点在348处),对susie序列图像中前11帧的统计表明,平均至少有24.6%的块用对偶法不能搜索到最小误差点.也就是说对偶法仅对特殊情况块才是正确的.如要匹配普遍准确,还需对该方法进行改进与扩展.

借鉴对偶算法搜索点数少和直线路径易于实现,速度快的优点,本文提出了搜索过程先粗后精的设想:利用已有的快速算法,在2个不同方向上去趋近最小误差点,然后利用全匹配方法估值准确的特点,再在此点周围逐点细化查找,实施分步匹配搜索.迂回逼近法体现了这一设想.算法示意如图2所示.

算法的过程描述如下:

1) 在搜索窗内的起始点 O 开始,先在横向(X 向)确定SAD值减小的方向,然后朝这方向连续搜索,直到不能减少为止,之后,再以此终点为始,确定纵向(Y 向)的搜索方向,继之朝该方向连续搜索,直到不能减少为止,得终点 A(如图2中D01所示);

2) 仍以 O 点为始,改变 X、Y 向的搜索次序,搜索过程同上,得终点 B(如图2中D02所示);

3) 以 A、B 二者的SAD值的较小点处为中心,用螺旋外延法[2]搜索匹配 n 周(本文使用3周),所得最小SAD点即为所求.

619	618	592	580	594	572	606	562	638	733
590	570	583	574	550	532	519	444	503	684
601	571	599	570	528	453	348	384	539	727
547	552	555	512	475	404	388	498	650	768
559	552	554	473	451	410	500	603	722	761
531	530	519	487	489	537	606	678	718	770
556	538	522	499	553	583	613	646	682	779
575	550	541	539	564	599	642	700	709	800
583	535	552	554	597	643	680	687	719	805
589	574	580	613	671	673	619	674	730	783

图1 匹配过程示例1(min_SAD=348)

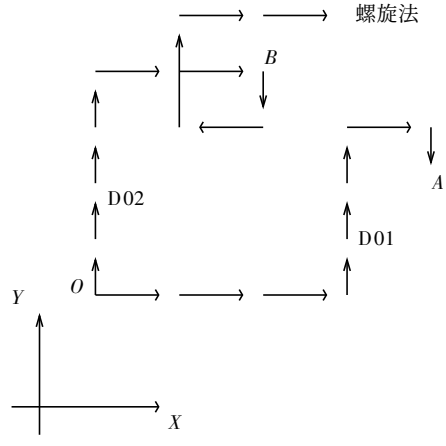


图2 迂回逼近法

图3示出了对一个宏块应用该算法的搜索匹配过程:自SAD=8964处始,经上述3步搜索得

7837	8124	8845	9774	10791	11839	12922	13997	15029
840	5373	5550	6529	7735	9018	10297	11552	12776
5210	4053	3212	3186	4404	5823	7306	8793	10225
5873	4536	3119	1930	1230	2756	4483	6130	7714
7269	6230	5347	4467	3445	2351	2725	4133	5661
9339	8964	8367	7538	6626	5410	3928	3662	4513
11855	11565	11093	10401	9578	8488	7106	5552	4983

图3 匹配过程示例2(min_SAD=1230)

最终估值点 1 230.

3.2　帧间匹配采用动态起始点

从上述的匹配例子可以看出, 初始的匹配点位置也很重要, 如果初始点本身选得距离实际匹配点较近, 用迂回逼近法就可以很快搜索到匹配点(例如在图 3 中假设能从 4 536 处开始) , 反之甚至可能匹配不到最佳点(假设从图 1 中 535 点处开始) . 为此在多帧共用一个基准帧连续匹配时, 对当前块的匹配, 用上一帧的对应块已经在参考帧中匹配到的估值块位置作参照, 依此点作初始搜索位置并定义搜索窗的大小, 其后各帧也均采用动态起始点的搜索. 方法是: 设本块位置为(c_X, c_Y) , 若前帧相应块的位移矢量为(v_X, v_Y) , 则在基准帧内以($c_X + v_X$, $c_Y + v_Y$) 重新定义搜索窗口并作为本算法的搜索起始点. 这样做后, 起始点距离匹配点可能更近, 既能提高匹配的速度, 又增大了匹配的可能性.

4　优化程序实现技术

好的算法只有在好的程序技术支持下才能获得最佳效果, 为此, 在迂回逼近法中对程序技术作了以下 2 个方面的优化工作.

4.1　定期比较过程 **SAD** 值

在已运算完的搜索块中有一个目前最小的 SAD 值 min _ SAD, 如果下一个搜索候选匹配块的 SAD 运算过程值已经超出 min _ SAD 时, 应该停止该块的匹配运算, 以节省时间. 但是在程序的什么位置进行判断比较合适, 有个整体效率问题. 本文改传统的逐点计算后立即比较判断的方法为宏块每行数据计算完后再比较判断的方法, 这一改进使速度得到进一步提高. 原因是判断语句也要占用时间, 如果在每个像素点运算后都执行判断, 就额外增加了和像素数数目同样多的判断语句, 即额外执行了许多指令, 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; 而以每一行数据作为计算单位时, 执行判断语句少, 且在每一个判断指令执行时, 当前块运算结果超出 min _ SAD 的概率远远超出前者, 付出的总代价小. 如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; 当满足条件时, 前者多用的判断语句个数等于计算过的像素点数, 而后者只等于行数(≤16) , 另至多比前者多计算了 15 个像素点. 全匹配的实验结果表明, 算法以行为单位相比以点为单位, 平均匹配时间快约 17. 3% .

4.2　用汇编语言实现核心运算

匹配算法中的核心运算是对每个搜索块计算 SAD 值, 这个计算的运算量是巨大的. 若搜索窗大小为 $W_M \times W_N$, 宏块为 16 × 16, 暂不调用绝对值函数运算和执行 if 判断语句的时间, 一幅图中仅求 SAD 的加减运算就有 HEIGHT/ 16 × WIDTH/ 16 × (W_M – 15) × (W_N – 15) × 16 × 16 × 2 次, 若以 352 × 288 图像大小, 32 × 32 搜索窗计, 需执行加法 5800 多万次; 估值出一个宏块的运动矢量也要执行加法近 15 万次. 在该情况下, 用 C 语言写成的传统程序, 对于块中数据的每一次 SAD 运算还存在不少低效操作, 如:

1)　用长指针或巨指针把指向数据读往内存时, 每次都要向数据段寄存器加载段值, 如果是巨指针则还要浪费用于格化的更多的时间, 仅就宏块大小的运算范围来说, 这部分操作有必要也有可能去掉;

2) 求像素灰度差之绝对值的运算在 C 中要调用算术函数, 函数本身要执行若干条指令才能完成, 而调用函数还要执行频繁的进栈出栈操作, 浪费了许多时间.

利用 C++ 与汇编的混合编程可以解决上述存在的问题. 其实现方法是: 机器的加减法指令操作默认数据的段值在数据段寄存器 DS, 而预测帧和当前帧的数据通常均在 64K 之外, 不能用一个段寄存器或固定的段值, 应在块匹配运算之前把 2 幅图中的宏块的段值分别赋予数据段寄存器和附加段寄存器, 段偏移值赋予不同的变址寄存器, 执行带段超越的加法操作. 由于块匹配运算期间段寄存器只装载一次, 所以机器的取数操作时间降到最少. 另外, 对于特定的 8 位无符号数(单波段灰度值) 减法操作, 分析其差值的绝对值运算有特殊的规律: 当无溢出时累加器的结果就等于绝对值; 而当出现溢出时也无需重新计算差值, 对该溢出的结果直接求补即为无符号数绝对值. 经过以上 2 个方面改进后, 既减少了数据的内存读取时间, 又使核心运算的指令数最少, 进一步提高了运算速度.

5　结　果

从 susie 标准序列图像中取出 11 幅 352 × 240 大小的图像, 宏块为 16 × 16, 搜索窗口的大小为 32 × 32(用 Pentium 100 微机做实验) . 为了比较匹

配算法在运动估值上的准确性,规定当前被预测帧的宏块全部用在预测帧所匹配到的最小误差宏块去替换,而无论大小.有关算法的匹配精度和匹配时间如表1所示. 其中,改进前的全匹配算法系传统顺序全匹配,改进后的全匹配算法包括了螺旋外延搜索及本文所述的程序优化技术;对偶法中也已采用了本文所述的程序技术.

表1 算法匹配结果

匹配方式	平均每帧估值时间/s	替换帧平均恢复信噪比/dB
全匹配(改进前)	40.11	35.885
全匹配(改进后)	1.11	35.885
对偶法	0.16	33.193
迂回逼近法	0.22	35.855

结果表明,迂回逼近法的匹配结果基本上达到了全匹配的精度和快速算法的速度.该算法除了具有一般快速算法搜索点数少和算法易于实现的优点外,更体现了搜索路径准确合理和从程序技术上更进一步提高算法速度的特色.本文提出的方法思想可以应用到相关的匹配算法当中.

参 考 文 献

1 林富宗,陆 达.多媒体与CD-ROM.北京:清华大学出版社,1995

2 何圣静.多媒体技术及其应用.北京:北京理工大学出版社,1996

3 Ghanbari M. The cross search algorithm for motion estimation. IEEE Trans on Commun, 1995, 38(7): 106~109

4 Walker D R, Rao K R. Motion-compensated coder. IEEE Trans on Commun, 1987, 35(11): 1171~1178

Design and Implementation of Fast Motion Estimation Algorithm

Liu Dechun Zhou Xiaokuan

(Beijing University of Aeronautics and Astronautics, School of Astronautics)

Abstract By sacrificing matching quality to reach higher matching speed is the common shortage of the published fast motion estimation algorithms. Based on the analysis of the strong and weak points of those motion estimation algorithms, a new fast motion estimation algorithm, using improved searching approach and optimized implementation techniques, is presented. In the new algorithm , the above-mentioned common shortage is eliminated. The speed of it can be compared with typical fast algorithm and the quality of it can be compared with the exhaustive searching algorithm. The principle, implementation technique and final testing results of the algorithm are also given.

Key words image processing; picture coding; integration; motion estimation