卫星软格式化同步器与资源卫星 1 号 02 星 CCD 数据移动窗相关技术研究 *

曹　宇[1,2]　杨仁忠[2]

[1](中国科学院研究生院　北京 100049)

[2](中国科学院中国遥感卫星地面站)

摘　要　处理速度一直是软件帧同步的瓶颈所在，通过引入 KMP 和 Rabin-Karp 算法，寻找到一种高效的帧同步处理算法，提高了处理速度，最终设计完成了通用的卫星软件帧同步器，并进行了相关验证测试和分析，给出了分析比较结果. 在分析中巴资源卫星 1 号 02 星数据格式的基础上，利用所设计的软件帧同步器和 DirectX 显示技术，实现了一套针对资源卫星双通道 CCD 传感器的全软件移动窗显示系统，并对该卫星数据的图像交叠等现象作了相应处理. 这套软件帧同步器和移动窗显示系统可以作为开发卫星接收记录系统的重要组成部分.

关键词　帧同步；移动窗；资源卫星；　CCD

中图法分类号　V 557

Software Frame Synchronizer and Related Technology Research of Moving Window Display for CCD Data of CBERS-2

CAO Yu[1,2]　YANG Renzhong[2]

[1](*The Graduate School of Chinese Academy of Science, Beijing* 100049)

[2](*The Remote Sensing Ground Station of China*)

Abstract　The processing speed is the key problem of software frame synchronization. This article, using the arithmetic of KMP and Rabin-Karp, give an effective method in software frame synchronization, which increase the processing speed. And a software frame synchronizer was built and related test and analysis were made, the result was given in the paper. With the analysis of the data format of CBERS-2 (China-Brazil Earth Resources Satellite) and using the software frame synchronizer and the technology of DirectX display, a software moving window display system of the CBERS CCD two channels sensor was made. The effect of the overlap data of this satellite was lighten by some method. The software frame synchronizer and moving window display system can act as very important parts in a data recording system.

Key words　Frame synchronize, Moving window, CBERS, CCD

---

* 中国科学院重大科学装置运行维护项目 ("小型地面接收站" 项目) 资助

2004-11-20 收到原稿，　2005-04-09 收到修定稿

1 引言

中巴资源卫星 1 号 02 星 (后文简称为资源卫星)4 个传输通道中的 2 个用于 CCD 相机的图像信息数据发送, 其载波频率为 8103 MHz 和 8321 MHz.

CCD 相机设置了 5 个波段: B1 (0.45—0.52 μm), B2 (0.52—0.59 μm), B3 (0.63—0.69 μm), B4 (0.77—0.89 μm), B5 (0.51—0.73 μm). CCD 相机 5 个谱段的图像数据分配到两个通道中, 一个 (CCD-DT1) 传输谱段为 2, 3, 4, 另一个 (CCD-DT2) 传输谱段为 1, 3, 5. 可以把每个通道传输的 3 个谱段命名为 Ba, Bb, Bc. 每个行周期 3 个通道按一定方式传输包含在一个扫描行内的 3 个谱段数据, 其格式如图 1 所示.

图 1 CCD 数据的基本格式

Fig.1 The basic format of CCD data

移动窗的实现可以是全硬件的, 可以是软硬件结合的, 也可以是全软件的. 本文即将讨论的移动窗实现方法是一种全软件的实现办法.

2 快速软格式化同步方法

使用软格式化同步相对于硬格式化同步的主要优点是开发成本低, 周期短, 而且可扩展性和可移植性比较好. 另外, 随着计算机计算速度和处理能力不断增强, 利用微机进行软格式化同步的条件已经具备.

一般的软格式化同步的方法是, 根据需要决定容错域值 $\delta \geqslant 0$, 逐位进行比对, 统计比对不相符的位的总数 Σ, 若有 $\Sigma \leqslant \delta$, 则认为找到同步码. 这种软同步的方法可以正确判别不超过 δ 位发生比特错误的同步码. 同时如果做一个数据中 0 和 1 分布均匀的假定 (不做特别说明以后均如此假定), 就可以计算出, 这种方法把数据段误识为同步码的概率为:

$$\frac{(C_{56}^2 + C_{56}^1 + 1) \times 2^8}{2^{64}} = \frac{1597}{2^{56}} = 2.21 \times 10^{-14}. \quad (1)$$

软格式化同步器的速度瓶颈在于同步的算法. 解决了速度问题, 软格式化相比于硬格式化就有了很大的优势.

KMP 算法是一种线性时间串匹配算法. 该算法主要的特点是无回溯和精确匹配. 无回溯这个特点可以大幅地提高比较速度. 将 KMP 等算法引入卫星软格式化同步技术手段, 目前还没有看到同类做法.

设计按位进行比较的方式. 把同步码看成是长度为 64 的模式串 P. 把读入内存的数据块看成是长度为内存数据块大小乘 8 的目标串 T. 其中 $P = p_0 p_1 \cdots p_{m-1}$, $T = t_0 t_1 \cdots t_{n-1}$. 首先根据失效算法对 P 生成长度为 64 的失效函数 F[i]=Fail(i). 生成失效函数依照下面的递推规则:

$$\mathrm{Fail}(j+1) = \begin{cases} \mathrm{Fail}^{(m)}(j) + 1, & k\text{为正整数}, \\ & p^{(k)}_{\mathrm{Fail}(j)+1} = p_j; \\ -1, & j = 0. \end{cases} \quad (2)$$

其中,

$$\mathrm{Fail}^{(m)}(j) = \mathrm{Fail}(\mathrm{Fail}^{(m-1)}(j)),$$
$$\mathrm{Fail}^{(1)}(j) = \mathrm{Fail}(j).$$

然后从 P 的第一位和 T 的第一位开始依次比较, 如果在某一趟比较中 P 的第 j ($j > 0$) 位失配, 那么下一趟比较时模式 P 的比较位置从 $p_{\mathrm{Fail}(j-1)+1}$ 开始, 而 T 的指针则不做回溯, 仍指向上一次比对失效的位置. 直到找到 T 串中与 P 串完全吻合的字串位置为止, 否则格式化同步失败. 匹配成功的时候可以直接根据从字节中提取该位时的位置判断出同步码在数据中相对于字节对齐偏移了多少位, 本算法给出的偏移位数是左偏的数值 (可取 0—7).

KMP 是精确匹配的, 但因为传输中可能产生误码, 就要求同步码搜索必须是带有一定容错性的. 这就需要对 KMP 方法进行修改.

修改的办法是采用 "首次出错回溯法", 以容错 2 位为例. 设定一个错位计数器, 并把它初始化为 0, 设定一个位置记录变量用来记录首次出错的位置. 当某次比对时, P 的第 j 位与 T 的第 k 位发生了比对失效, 则把 k 送位置记录变量, 继续进行比较, 同时容错计数器加 1.

以后再次出现比较结果不一致的时候, 先把容错计数器加 1, 然后比较容错计数器是否不大于设定的容错域值 (比如 2), 是则继续进行比较, 否则认为同步错误, 结束本趟比较. 跳到 T 串第 k 位, 从 P 串的 $p_{\mathrm{Fail}(j-1)+1}$ 位置重新开始比较. 直到某次比

较的结果使得 P 串比较完毕而容错计数器仍然不大于容错域值即认为找到了同步码. 根据上面给出的算法将在比对发现第 3 个错误时停止比较进行回溯.

如果假设每一位发生错码的概率是 p, 则第一个出错的位置的数学期望值为

$$\frac{\sum_{n=1}^{64} np(1-p)^{n-1}}{\sum_{n=1}^{64} p(1-p)^{n-1}}, \quad (p \neq 0). \tag{3}$$

设这个数值为 S, 则当 $p = 0.1$ 时, $S = 10.01$; $p = 0.001$ 时, $S = 32.16$; $p \to 0$, $S = 32.5$.

第三个位置错误的数学期望值可以近似地认为是上式的 3 倍, 即

$$3 \times \frac{\sum_{n=1}^{64} np(1-p)^{n-1}}{\sum_{n=1}^{64} p(1-p)^{n-1}}, \quad (p \neq 0). \tag{4}$$

也就是按照数学期望, 容错两位花费的时间约是无容错的 3 倍左右. 可以看出, 既然要增加容错, 就必须引入回溯, 引入了回溯必然会以牺牲一定的速度为代价. 在容错两位的情况下, KMP 算法把数据段误识别为同步码的概率和 (1) 式给出的结果是相同的.

既然引入回溯导致了速度的下降, 那么是否有其他方法可以加快搜索速度, 比如能在做定量的比对之前作一个定性的分析, 为此找到了 Rabin-Karp 算法.

Rabin-Karp 算法的基本思想是对一个长度为 m 的字符串赋予一个 Hash 函数, 那么就只有那些和模式 P 具有相同 Hash 函数值的字串才有可能模式匹配. Hash 函数的本质就是散列, 即通过一个变换实现数据和地址的对应. 但是不管怎样的散列, 都不可能完全避免冲突, 或者说为了完全避免冲突所付出的代价是不划算的. 这其实提出了格式化同步的一个必要条件, 也就是可以在 KMP 之前就排除掉绝大部分长度和 P 相同的字串, 从而大大减少比对次数达到提高速度的目的.

散列的方法有很多, 求模取余是一种比较常见的方法. 如果一串字串中出现的字符集是 Σ, 令 $S = |\Sigma|$, 那么把 P 看成是 S 进制数就可以把 P 数值化. 若 $P = b_1 b_2 \cdots b_m$, 有

$$b_m + b_{m-1} s + b_{m-2} s^2 + \cdots + b_1 s^{m-1}.$$

取其 Hash 函数:

$$\beta = b_m + b_{m-1} s + b_{m-2} s^2 + \cdots + b_1 s^{m-1} \bmod q.$$

其中 q 是一个素数.

对于同步码来说显然 $S = 2$. 但根据 Hash 函数, b_i 有不同的权重, 对于考虑误码容错将很不方便, 这意味着虽然同样是一位错, 第 2 位错和第 63 位错对结果带来的影响将是截然不同的.

把问题更加简化一下, 令 b_i 都是等权重的, 这样简化的 Hash 函数不仅计算起来方便, 而且容错性能也相当好, 只要规定 T 串所有长度为 64 的字串中, 只有那些简化的 Hash 函数的数值和 P 相吻合, 或者差的绝对值不大于容错域值的字串才进行 KMP 比较. 就可以进一步提高比较的速度.

对于数据中的滑帧现象, 从实际接收情况来看, 发生的概率远比误码低, 且多发生在信号受干扰较强的时候. 目前部分实际运行系统对滑帧的数据是直接丢弃, 并在移动窗显示的时候对这一部分做一些插值等处理. 因此本文的算法并未针对滑帧做专门讨论, 但就本算法的改进而言可以从两个角度考虑解决滑帧问题. 一方面对于定帧长的卫星, 如资源卫星, 可以在距离上一个同步码字一个帧长的位置上进行搜索, 另外上面所提供的可容错误码的 Hash 函数实际上也可以容错同样位数的滑帧, 所以仍然可以保证高效.

通过软件编程可以实现上述软格式同步的测试. 测试方法是, 在内存生成的数据, 长度是 125MB (1000Mb), 同步码位于数据末尾, 同步码的偏移为 0—7 的随机数, 比对长度 64 位, 容错误差 2 位. 测试环境是 PIV-2.4 双 CPU 和 1 GB 内存 (测试结果见表 1).

在操作系统中, 包含有高精度运行计数器, 可以获得高精度定时间隔, 其精度与中央处理器时钟频率有关, 测试时间选用了调用读取 CPU 时间戳 API 的办法, 用这个方法时间可以达到的精度大约是 10 ms. 在 PIV-2.4 GHz×2, MEM-1 GB 平台上的测试结果可以看出, 首先这个算法时间复杂度是 $O(n)$, 即搜索时间和数据量呈线性关系, 其次这个算法搜索速度比较稳定, 最高速度 1084.6 Mbps, 最低速度 1066.1 Mbps, 平均可以达到 1070.2 Mbps.

表 1 软格式化同步快速算法实验结果
(模拟资源卫星数据)

Table 1 The result of software frame synchronization test (Simulated Data)

计算机平台 PIV-2.4 GHz×2/MEM-1 GB/Windows 2000		
试验编号	同步码偏移	所耗时间/s
1	1	0.937
2	2	0.937
3	1	0.937
4	6	0.938
5	4	0.937
6	4	0.938
7	3	0.938
8	7	0.922
9	5	0.922
10	0	0.938

由于同步码在数据流中往往不是以字节形式对齐的，所以必须对数据块进行移位处理，达到字节对齐的目的. 上面设计的同步算法中给出的同步码偏移量是左偏的数值，也就是对数据段要进行右移操作. 为了达到快速移位的目的，在处理之前建立一个查找表，表中存储各种情况字节移位的结果，根据同步码偏移量是 syn_offset 和当前没对齐的两个字节就可以直接确定字节对齐后的结果.

3 多传感器数据交叠处理方法

包括资源卫星在内的一些卫星，数据是由多个传感器拼接而成的，这些传感器采集到的信息有一部分是重叠的，如图 2 所示.

峰值的高低代表了图像感光的强弱，在两个传感器交接处，一个传感器光渐弱，另一个渐强最终完成交接. 交接时的数据两个传感器均作了记录.

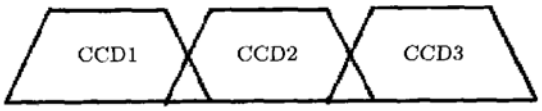

图 2 传感器之间数据交叠的状况
Fig.2 The data overlap of the sensor

数据下传的顺序如图 3 所示.

多个传感器是依次下传的，从而在传感器交接的地方产生图像的不连续和黑影. 为了保证做出的移动窗图像美观连续必须进行处理.

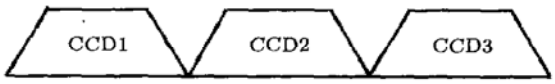

图 3 传感器下传的数据
Fig.3 The Down-Link data of the sensor

资源卫星上，两个传感器之间的交接采用了全反全透棱镜的控制方式，也就是两个传感器分别接受了透射光和反射光，那么最简单的办法就是对光强进行叠加.

RGB 方式三个分量所表示的数值就是该种光的强度. 可以认为反射光和透射光都是色彩不饱和的光. 假定反射光色饱和度是 α, 则同一点的透射光色饱和是 $(1-\alpha)$. 假定反射光饱和值为 A, 透射光饱和值为 B, 那么两种光的光叠加应为

$$A \times \alpha + B \times (1-\alpha). \tag{5}$$

α 虽然是一个随位置变化的数值，但是对同一位置而言，公式 (3) 中 $A = B$, 而且 $A \times \alpha$ 和 $B \times (1-\alpha)$ 分别就是对应的 RGB 值. 因此光强叠加实际上就是同一点的 RGB 数值直接叠加.

光强上的叠加可以去掉接缝，但由于各个 CCD 的不同，直接光强叠加图像虽然连续，但往往还会产生纵向条纹.

减轻纵向条纹我们采用了亮度补偿法. 首先认为卫星图像是连续的，那么先计算出交接处之前的光亮度，再计算出交接处作了光强叠加的亮度，对比两者差值，对交接处进行补偿，可以使纵向条纹变得暗淡.

光亮度的计算方法也就是灰度计算方法，如果把灰度分为 255 个等级那么可以按照下式计算

$$\lfloor 0.299 \times R + 0.587 \times G + 0.114 \times B \rfloor. \tag{6}$$

其中 $\lfloor\ \rfloor$ 表示取整.

4 基于 DirectDraw 移动窗显示设计

DirectX 软件开发工具包 (SDK) 提供了一套应用程序编程接口，它可以提供开发高质量、实时的应用程序所需要的各种资源.

DirectDraw 是 DirectX SDK 中最主要的一个部件. 它允许程序员直接地操作显存，硬件位图映射，以及硬件覆盖和换页技术.

利用 DirectDraw 的 Blit 和翻页的特性，画面的移动显示的处理方法如下.

①设置一个主页面、后台缓冲区、离屏页面 1、离屏页面 2. 设置参数 IsFirstScreen 表示目前显示的是不是第一屏的画面， ImageLine 向离屏页面写入了多少行数据， HEIGHT 为页面的高度.

②若 IsFirstScreen=1，即目前显示第一屏画面. 此时只要向离屏页面 2 写入数据，利用换页显示在屏幕上. 当 ImageLine 增加到后台缓冲区的高度值时，第一屏画面显示结束，令 IsFirstScreen=0. ImageLine 重新设置为 1，离屏页面 1 与离屏页面 2 的指针互换.

③此时离屏页面 1 中存储着刚才的第一屏画面. 向离屏页面 2 写入一行数据后，做两个页面部分数据的一个交换操作. 然后 ImageLine 自增 1. ImageLine 增加到后台缓冲区的高度值时，重新设置为 1，离屏页面 1 与离屏页面 2 的指针互换.

④页面每刷新一行数据，就进行一次换页操作. 每屏写满后，页面 lpDDSPic1、lpDDSPic2 的指针互换.

5 测试效果

本文数据源来自 RAID 磁盘阵列存储的资源卫星的原始数据. 程序执行过程是：开始回放时，接收数据线程每次从 RAID 读入 256 KB 的数据，读取完毕后通知数据处理线程接收到数据，数据处理线程对这块数据进行软格式化同步，数据解扰，接

图 4 资源卫星 CCD1 原始数据回放移动窗

Fig.4 The moving window display of the CCD1 original data of CBERS

缝处理，提取 RGB 颜色进行彩色合成并进行移动窗显示等操作，操作完成后通知接收数据线程可以开始接收新的数据.

资源卫星是双通道的，表 2 列出了只处理一个通道和同时处理两个通道的处理速度结果.

表 2 软件移动窗处理速度测试

Table 2 The process speed test of software moving window display

类别	编号	数据量/Byte	处理时间/s	处理速度/Mbps
单通道	1	2 296 680 740	55	318.6
	2	2 296 680 740	58	302.1
	3	2 453 024 840	59	317.2
	4	2 453 024 840	59	317.2
双通道	1	4 749 705 580	133	272.5
	2	4 749 705 580	124	292.2
	3	4 749 705 580	122	297.0

注：双通道的数据量指的是两个通道数据量之和.

图 4 是资源卫星的 CCD 原始数据回放的效果图，其中纵向仍可以看到接缝的位置，但比处理之前已经很不明显了，目前实际运行的系统这个接缝处理后也还是可以看得到的.

6 结论

本文针对遥感地面系统的格式同步和移动窗显示这一关键技术环节进行了处理算法的研究. 通过对资源卫星 1 号 02 星的数据帧格式的分析，将 KMP 和 Rabin-Karp 算法引入到格式同步处理技术中，实现了一套软件格式同步器-移动窗的系统，并在实测中得到了较好的效果.

参考文献

[1] Yin R K, Tao Y L, Xie R Y, Sheng X H. Data Structures (Expatiate with OOP method and C++). Beijing: Tsinghua University Press, 2002. in Chinese (殷人昆, 陶永雷, 谢若阳, 盛绚华. 数据结构 (用面向对象方法与 C++ 描述). 北京: 清华大学出版社, 2002)

[2] Kruglinski D J. Inside Visual C++, 4th Edition. Beijing: Tsinghua University Press, 1999, in Chinese (Kruglinski D J. Visual C++ 技术内幕第 4 版. 北京: 清华大学出版社, 1991)

[3] Lu K C. Computer Arithmetic Introduction-Design and Analysis. Beijing: Tsinghua University Press, 1996. in Chinese (卢开澄. 计算机算法导引-设计与分析. 北京: 清华大学出版社, 1996)

[4] Jiang W P. Improvement of KMP scan algorithm. *J. Guangxi Academy Sci.*, 2001, **17**(3):133—136. in Chinese (蒋文沛. KMP 扫描算法的改进. 广西科学院学报, 2001, **17**(3):133—136)

[5] Li J. The arithmetic of matching the string's mode——talking over the KMP arithmetic. *J. Qingdao Ins. Chem. Tech.*, 2002, **23**(2):78—80. in Chinese (李静. 字符串的模式匹配算法 —— 基于 KMP 算法的讨论. 青岛化工学院学报, 2002, **23**(2):78—80)

[6] Yao L H, Zi X C, Zhang S L, Xie L. An efficient multiple strings matching algorithm. *Comp. Appl. Soft.*, 2004, **21**(1):7—9. in Chinese (姚立红, 訾小超, 张绍莲, 谢立. 一种高效的多目标串匹配算法. 计算机应用与软件, 2004, **21**(1):7—9)