高技术通讯 2012 年 第22卷 第10期:999 ~ 1006

doi:10.3772/j.issn.1002-0470.2012.10.001

软件 GPS 接收机跟踪阶段的自适应帧选择算法①

姚相振②*** 崔绍龙*** 方金云*

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院研究生院 北京 100049)

摘 要 从接收机的实际应用模式出发,以提高接收机初次定位的效率为目的,提出了软件 GPS 接收机跟踪阶段的一种自适应帧选择算法。该算法通过先跟踪并解调一段导航数据来判断初始帧的帧号,然后制定相应的策略有选择地处理接下来的导航数据,舍弃与初始定位无关的帧,从而大大减少跟踪阶段的数据处理量,提高初次定位效率,缩短冷启动时间(TTFF)。该算法是从原始数据源出发来减少跟踪环路的数据处理量,因此能显著缩短 TTFF 的平均时间。实验结果表明,此算法能有效减少跟踪操作,大幅提高初次定位效率,为后期嵌入式平台上的 GPS 实时纯软件基带信号处理奠定了基础。

关键词 软件 GPS 接收机,初次定位,信号跟踪,自适应帧选择

0 引 言

由于传统的硬件 GPS 接收机在成本和灵活性上不具优势,随着芯片制造技术的发展和嵌入式移动终端计算能力的提升,诞生了一种新的接收机体系结构——基于软件无线电的 GPS 接收机结构[1]。与传统的硬件接收机相比,软件 GPS 接收机的处理和运算功能都在可编程的微处理器中实现,使系统具有高度的灵活性。当需要对不同的算法进行测试以及 GPS 信号发生变化时,这种灵活性就会体现出来,此时,只需要非常简单的更换程序,而并不需要对系统结构做太大的调整就可以实现和适应新的功能。软件接收机的这种可以通过软件的升级获得新功能的特性,将具有更广阔的应用空间[2]。

在嵌入式软件无线电 GPS 接收机的应用中,面临的最大挑战就是资源与运算能力的限制,因此如何在嵌入式移动设备上实现实时处理,则成为软件接收机系统的当务之急。在整个接收机的处理过程中,最耗时的部分是卫星信号的跟踪环路,在 1GHz ARM-Cortex-A8 处理器以及嵌入式 Ubuntu-Linux 操作系统的试验平台下,跟踪阶段采用双通道并行跟踪策略,此阶段的耗时占整个基带信号处理时间的 70% 左右,因此,提高跟踪阶段的效率成了实现实时性的关键。目前提高跟踪操作效率的方法主要集中在改进跟踪环路的设计,降低载波跟踪环以及码跟踪环的复杂度上[3-5],另有一些方法通过并行的方式提高跟踪效率,包括多核多线程的并行算法,以及其它形式的并行模式[6]。为了完成 GPS 接收机的初次定位,必须获得 4 颗以上的卫星星历数据,而这些包含卫星轨道位置数据的星历分布在导航数据的前 3 个子帧当中,为了接收 1、2、3 子帧的完整数据,传统的方法是跟踪和解调全部 5 个子帧,由于接收机接收信号时刻的随机性以及跟踪环路的锁定时间因素,解调出的首个子帧可能存在不完整性,实际必须跟踪 6 个子帧才能确保其中包含完整的 1、2、3 子帧数据,这无疑加重了跟踪环路的处理量。为了减少跟踪环路的这些冗余处理,我们根据 GPS 卫星导航数据帧结构的特点,提出了一种自适应选帧算法,该算法能在确保获得前三个子帧数据的前提下,尽量减少跟踪环路处理量,有效缩短冷启动时间(time to first fix,TTFF),提高初次定位效率。

1 导航数据帧结构分析

GPS 导航数据有其固定的电文格式,即 5 个导航数据子帧组成一个完整的导航页,所有子帧的前

① 863 计划(2009AA12Z315)资助项目。

② 男,1984 年生,博士生;研究方向:软件 GNSS 接收机关键技术研究;联系人,E-mail: yaoxiangzhen@ict.ac.cn

(收稿日期:2011-07-01)

两个字都是相同的,分别是 TLM(telemetry,遥测)字跟交接字(hand over word,HOW),然后每个导航字包含 30 比特的二进制数据,并且消息是从第 1 比特到第 30 比特传递的,这两个字的描述见图 1。TLM 字的开头是一个 8 位的前导码,接下来是 16 位的保留位和 6 位的奇偶校验位。前导码的格式将会用来校验导航数据以检测子帧的开始[7]。

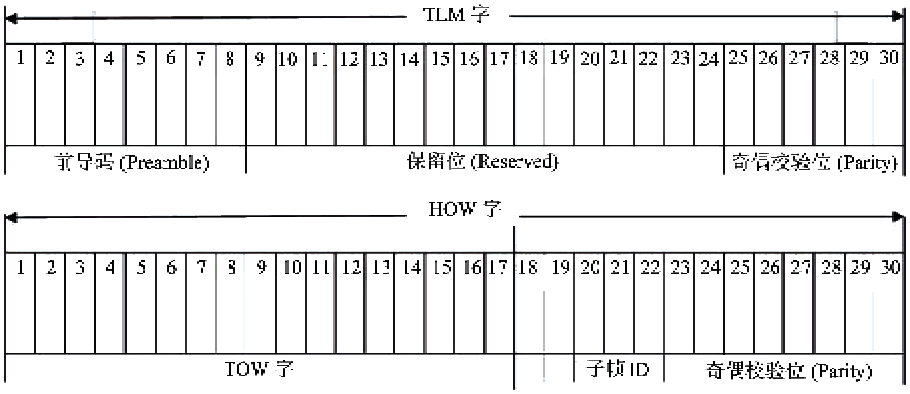

图 1 导航数据帧结构图

HOW 字被分为以下 4 部分:

a) 前 17 比特是 TOW(truncated time of week)字,它提供了以 6 秒位单位的时间计数。

b) 接下来的 2 比特(18,19)是旗帜位,对于卫星 001 配置(block II 卫星)第 18 位是一个警报位,第 19 位是反欺骗位,大多数 block I 卫星都是实验型的,所有在轨卫星都来自 block II,当第 18 位等于 1 时意味着卫星用户的位置精度可能尚不如子帧 1 预示的准确,第 19 位等于 1 意味着反欺骗模式打开。

c) 接下来的 3 位(20-22)是子帧 ID,它们的值分别是 1,2,3,4,5(001,010,011,100,101),以用来标示是第几帧数据。这些数据将会用来做子帧校验。

d) 最后的 8 位(23-30)将会用来做奇偶校验。

2 导航数据解调和帧结构分析

导航数据的解调,是将数据从接收信号中分离、提取出来的过程。这个过程包括数据解调、位同步、帧同步、校验等。当 C/A 码跟踪环路锁定时,从同相臂的瞬时支路就可以解调出包含完整数据的二进制码流。这个二进制码流就是最原始的卫星导航数据。

2.1 比特同步

在跟踪环锁定的状态下,环路的输出为 1bit/ms。而导航数据的比特长度是 20ms,因此每 20 个输出数据要转化为 1 个导航数据,也就是完成从 1kHz 的数据变换到 50Hz 的数据。关键的问题是找到正确的导航数据位的起始和结束边界,这个过程叫比特同步。

同步方法:通过寻找比特跳变的位置来确定。利用导航数据每 20ms 才可能发生一次跳变的规律,很容易识别出所有的跳变位置。通过过零检测法把跟踪环路的输出值转换为 0 和 1 的序列,当跟踪环此毫秒输出大于零则转化为 1,否则转化为 0,然后比较前后两个值的差,如果不相等则判断数据跳变位发生[8]。本文采取的方法是从第一个数据跳变位开始记录,如果后面的一段数据中隔 20ms 的整倍数的位置同样出现跳变位则计数加 1,在此段数据中计数没有加到 3 则选取下一个跳变位继续循环计数,等计数达到 3 时我们暂且认为此位就是数据的比特起始位,如果在后面的校验过程中出现差错则返回该程序继续循环查找,此时,本次循环的累加器必须加 1 达到 4 才能跳出程序继续运行,以此反复执行,每次累加器上限都加 1,直到校验成功。

2.2 帧同步

根据帧同步头的特殊标志,加上子帧长度校验和子帧的奇偶校验,就可以完成帧同步过程。

首先查找帧开头的前导码(10001011),该码是典型的巴克码。因为还没有做奇偶校验,因此相反极性的前导码(01110100)也需要被检测,根据子帧的固定长度,连续检测相隔一帧长度的数据,以确定帧同步的正确性。

帧同步完成以后需要做奇偶校验:一个导航字有 30 个位,其中 6 位是奇偶校验位,这些奇偶校验位用来做奇偶校验以更正导航数据的极性。如果奇偶校验失败的话,导航数据将不能被使用,为了做奇偶校验,将使用 8 个奇偶校验位,额外的这 2 位来自上一个导航字的最后 2 位。

以 D_i 代表从接收机得到的导航字的一个位,$i=1,2,3,\cdots,24$ 代表导航数据,$i=25,26,\cdots,30$ 代表奇偶校验位,经过一系列特定的模二加运算得到一组新的 D_i,$i=1,2,3,\cdots,30$,D_{25} 到 D_{30} 为奇偶校验数据。

在 GPS 接收机中,由于载波采用 BPSK 调制,相干解调的结果会造成载波相位模糊,使得导航数据极性不确定,为了纠正这种不确定性,在奇偶校验的过程中,根据前一个字的 D_{30} 位来确定后一个字前 24 位的极性,如果前一个字的 D_{30} 位是 1,那么接下来的 24 位就要转换极性,如果是 0,则保留原来的极性。在本文中,为了把模二加运算转换为乘法运算,将 0 转化为 1,将 1 转化为 -1,这样的操作可以扩展到多个输入的情况,等计算完成以后再将 1 和 -1 转化为 0 和 1,完成数据的预处理[9]。

2.3 帧结构分析

GPS 导航电文可以分为 3 个数据块,第一数据块只包含在第 1 帧,参数包括周数、测距精度、卫星健康状态字、电离层延迟参数,L2 的调制波类型以及卫星时钟修正参数 t_{oc},a_{f0},a_{f1},a_{f2}。

第二数据块包括第 2 和第 3 帧,包含信号传送卫星的星历,这些数据提供了十分精确的卫星轨道信息:

(1)开普勒轨道 6 参数:

M_0:参考时刻 t_{oe} 的平近角点

$\sqrt{a_s}$:卫星椭圆轨道的半长轴的平方根

e_s:卫星椭圆轨道的偏心率

i_0:参考时刻 t_{oe} 的轨道倾角

Ω_0:参考时刻 t_{oe} 的轨道升交点准精度

ω:轨道近地点角距

(2)摄动轨道 9 参数:

Δn:平均角速度的修正

Ω:升交点赤经变化率

i:轨道倾角变化率

C_{rs}:升交角距的余弦调和项改正的幅度

C_{us}:升交角距的正弦调和项改正的幅度

C_{ic}:轨道倾角的余弦调和项改正的幅度

C_{is}:轨道倾角的正弦调和项改正的幅度

C_{rc}:轨道半轴的余弦调和项改正的幅度

C_{rs}:轨道半轴的正弦调和项改正的幅度

第三数据块包含第 4 和第 5 子帧,这两个子帧都各自再换接 25 次,主要向用户提供 GPS 卫星的概略星历及卫星工作状态的信息,所以称为卫星的历数。当用户捕获一颗卫星后,便可从其导航电文的第三数据块中得知其他卫星的概略位置、卫星时钟的概略修正数以及工作状态等信息。这对于选择适宜的观测卫星以构成最佳几何分布来提高定位精度具有重要意义。用于初次定位所需要的参数都包含在第一和第二数据块中,即前 3 子帧数据,因此,通过预处理一小段数据来判断帧的排列情况,以时间以及资源消耗最小的自适应帧选择算法来迅速提取出导航数据的前 3 子帧来实现快速初次定位是本文的主要工作。

3 自适应帧选择算法

在传统的 GPS 接收机中,要把全部的 6 帧数据都解调出来再进行接下来的位置解算,这样做的优点是对接收到的导航数据做统一处理,比较简单,缺点是增加了接收机进行初次定位的时间消耗,因为单纯依靠前 3 帧的数据就可以进行初次定位,后面两帧导航数据用来做下阶段卫星可视预测。接收机进行初次定位的快慢也是衡量一个接收机性能的关键指标,自适应帧选择算法在额外时间消耗可忽略不计的情况下动态选取导航数据的前 3 帧进行组合来实现快速初次定位。

3.1 最小代价帧号抽取法

在帧同步的过程中,我们采取了一种低信噪比下的容错方法:首先,不进行 1kHz 的数据到 50Hz 的数据的转换,而用原始的数据进行帧起始位置的判断,这样前导码(10001011)或(01110100)的每位将被拉长至 20 倍长度,形成一个 160 位的前导码,然后跟原始导航数据做相关运算,对应位置相同则计数加 1,计算出累加和,当累加和大于某个阈值时则可判定为匹配成功。本文在进行大量实验后选取阈值为 153,这样做的效果使得一些信噪比比较低的导航信号同样可以匹配成功以进行后续操作[10]。

首先要跟踪一段导航数据,然后解调数据,提取帧号,根据此帧号来制定接下来的跟踪策略。要保证一次性读取出帧号,至少要跟踪并解调一个完整子帧长度的信号,但是考虑到在帧同步的过程中,为了确定子帧的起始端需要做多帧等间隔的比对,以

确定起始端。本文采取的策略是先进行两帧的起始端比对,然后再通过奇偶校验位进行附加判断,综合两个条件,初次跟踪解调两个子帧长度数据,在原始的导航数据下(从 RF 前端 GP2015 获得),为 12000 位数据。

在帧同步过程中,只对两帧数据的起始端进行比对,然后根据其它奇偶校验位进行二次判断,如子帧中 HOW 字的最后两位总是 0,若比对成功,则结合抽取的帧号进行最终判断,帧号转化为十进制应为 1 到 5 之间,这样在分保证了帧同步正确性的前提下进行了开销较小的两帧数据处理。

3.2 初始帧号提取后的动态响应措施

在帧同步处理完之后提取出第一个子帧的帧号,然后根据帧号进行相应的处理,此时,可能会出现图 2 和图 3 所示的两种情况。

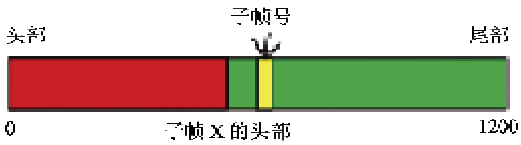

图 2 *BeginIndex* 大于等于 5000

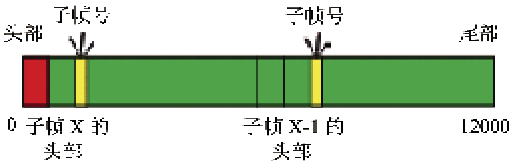

图 3 *BeginIndex* 小于 5000

在这两个图中,子帧 X 的头部位置标记为 *BeginIndex*,之前的部分则表示锁定子帧起始端前的数据,这些数据在下一步解算过程中将被抛弃。在图 2 中,此类情况属于接收机接收数据的时刻已经错过了第一个子帧的帧头,无法获得帧起始端,从而使得这个子帧成为报废数据,接下来程序检测到第二个帧的起始端,从此处开始保存有效数据进行后续处理。图 3 的情况属于程序很快检测到了第 1 个子帧的起始端,开始保存有效数据,接着又检测到第 2 个帧的起始端,在这种情况下丢弃的数据很少,处理效率比较高。当接收的数据的开头错过了子帧帧号的位置则后面的大部分数据都将被抛弃,帧号在一个子帧的第 50、51、52 位(见图 1),也就是原始数据的 1000 至 1040 位,一个子帧的长度对应于原始数据的长度为(10 字 ×30 位 ×20) =6000 位,因此当返回的有效数据起始端位置 *BeginIndex* 值大于(6000 - 1000) =5000 时属于第一种情况,此时只能通过特定位置的奇偶校验位以及帧号十进制数的范围来确定子帧的起始端,当小于 5000 时属于第二种情况,然后结合这两种情况和获得的帧号制定接下来的处理策略。

3.3 算法描述

定义 1.

(1) *BeginIndex* 为初次检测到的有效数据的起始端位置, *FrameID* 为初次检测到的子帧帧号, *Check _ result* 为检测结果,成功为 1,失败为 0;

(2) $P(i,j)$ 代表导航数据第 i 子帧的第 j 位, $L(j)$ 代表本地前导码的第 j 位,如前所述, $L(j)$ 共 160 位。

定义 2.

函数 1. *read*(m,n) | 从 m 点开始读取并处理 n 位数据; | 。

函数 2.

```
NormalCheck( )
Check _ result = Check(frame1 , frame2) ;
  if Check _ result = =0
      {k = BeginIndex + 1;
     Goto Step1 ; }
```

在函数 2 中进行的是常规的检测,当初次检测到第一个子帧的起始端后,用相隔一个子帧长度的 160 位数据与其做相关,相关的结果如果大于阈值则表明判断成功,否则重新返回 Step1 寻找起始端, k 为 Step1 中式(1)的积分起始点索引。

函数 3.

```
ParityCheck( )
Check _ result = Check(frame1 , frame2) ;
  if Check _ result = =0
      {k = BeginIndex + 1;
         Goto Step1 ; }
  else
      {  read(12000, 1040) ;
       NormalCheck( ) ;}
```

函数 3 是利用奇偶检验位来辅助进行检测,当检测成功后继续读取 1040 位导航数据以保证能解调出第 2 个子帧的头部以及帧号,然后再进行一次常规检测,做两个子帧前导码的比对,匹配失败则重新返回到 Step1 寻找起始端。

Step1:读取并跟踪处理两个子帧长度的数据,共 12000 位,完成后用下述方法直接进行帧同步操作,省略位同步操作:

$$A = \sum_{j=k+1}^{k+160} P(i,j) \times L(j) \quad (1)$$

初始值 $k=0$，当 $A \geqslant 153$ 或 $A \leqslant -153$ 时(由于极性的随机性会造成符号相反情况出现)，检测到子帧起始端;否则，$k = k+1$ 继续检测，直到检测成功。检测完成后得到 *BeginIndex* 值，并提取出子帧帧号 *FrameID*，当 *BeginIndex* 小于等于 5000 时，进入 Step2，否则进入 Step3。

Step2.

BeginIndex < =5000

```
if (FrameID = =1){
    NormalCheck();
    read(12000, 6000 + BeginIndex);}
if (FrameID = =2){
    NormalCheck();
    read(12000, BeginIndex);
    read(24000 + BeginIndex, 6000);}
if (FrameID = =3){
    NormalCheck();
    read(12000 +6000 + BeginIndex,12000);}
if (FrameID = =4){
    NormalCheck();
    read(12000 + BeginIndex, 18000);}
if (FrameID = =5){
    NormalCheck();
    read(12000, 12000 + BeginIndex);}
```

Step3：

BeginIndex >5000

```
if (FrameID = =1){
    ParityCheck();
    read(12000 + 1040, 4960 + BeginIndex);}
if (FrameID = =2){
    ParityCheck();
    read(13040, BeginIndex-1040);
    read(13040 + (BeginIndex-1040) + 12000, 6000);}
if (FrameID = =3){
    ParityCheck();
    read(13040 + (4960 + BeginIndex), 12000);}
if (FrameID = =4){
    ParityCheck();
    read(13040 + (BeginIndex-1040), 18000);}
if (FrameID = =5){
    ParityCheck();
    read (13040, 10960 + BeginIndex);}
```

在上述方法的 Step2 跟 Step3 中，当 *FrameID* = 2,3,4 时需要跳过与初次定位无关的子帧(见图 4)，灰色网格部分为跳过的数据，这种情况下还需要做一些细节上的处理，在跳帧时应该考虑到跟踪环路锁定延迟的问题，跟踪环路只有在锁定后才会保持稳定状态，在此之前的状态比较紊乱，此时的输出不能作为解调数据使用，因此在跳帧时应在下一整页开始前多读取一小部分数据。

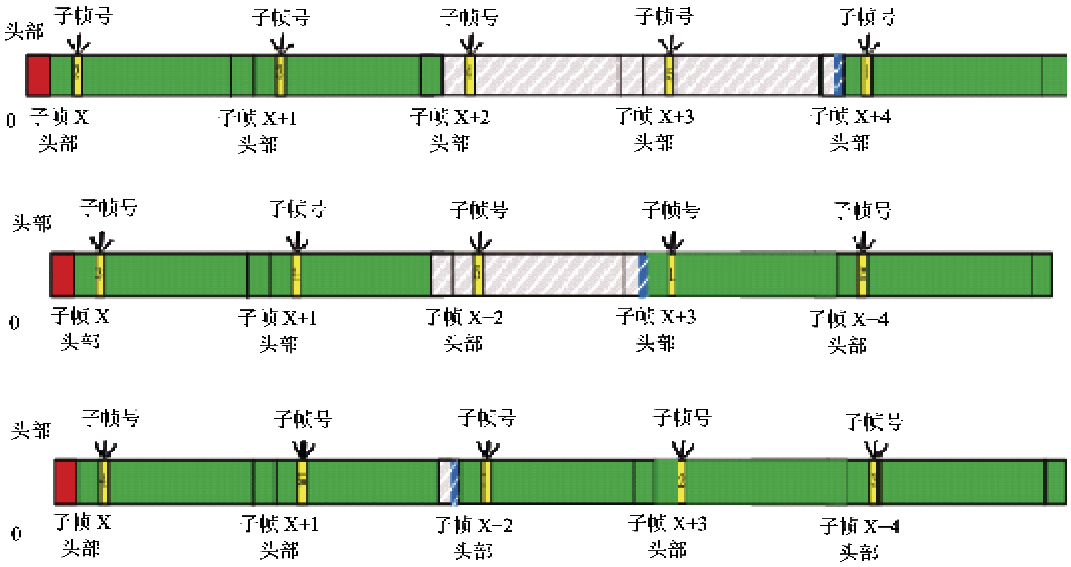

图 4　跳帧情况处理示意图

3.4　锁相环捕获时间分析

对于 GPS 软件接收机的锁相环路，频率捕获要比相位捕获慢得多，且更复杂，因此其总的捕获时间由频率捕获时间来决定。本文的跟踪混路采用传统

的二阶锁相环,其传递函数为

$$H(S) = \frac{S(2\zeta\omega_n - \frac{\omega_n^2}{K}) + \omega_n^2}{S^2 + 2\zeta\omega_n S + \omega_n^2} \quad (2)$$

其中 K 为环路增益, ω_n 为无阻尼固有频率, ζ 为阻尼系数, S 为拉普拉斯变换的独立变量。一般情况下 K 远大于 ω_n, 因此上式近似理想二阶环:

$$H(S) = \frac{2\zeta\omega_n S + \omega_n^2}{S^2 + 2\zeta\omega_n S + \omega_n^2} \quad (3)$$

其中 $\omega_n = \sqrt{\frac{K}{\tau_1}}$, $\zeta = \frac{\tau_2}{2}\sqrt{\frac{K}{\tau_1}}$。$\tau_1$ 和 τ_2 为时间常数。

本文为了提高锁相环路的捕获性能,在导航数据进入环路之前通过缩小载波频率误差,将频差落于环路的快捕带

$$\Delta\omega_L = K\frac{\tau_2}{\tau_1} = 2\zeta\omega_n \quad (4)$$

内。捕捉带为

$$\Delta\omega_p = 2\sqrt{K\zeta\omega_n} \quad (5)$$

最大捕获时间为

$$T_{Lmax} \approx \frac{5}{\zeta\omega_n} \quad (6)$$

本文载波跟踪的环路噪声带宽 LBW =25Hz,阻尼系数 ζ 为0.707,由 $\omega_n = \frac{LBW \times 8 \times \zeta}{4 \times \zeta^2 + 1}$ 得出 ω_n = 47.1728(rad/s),带入式(6)得出 T_{Lmax} =150.02ms,因此环路的捕获时间大约为150.02ms,对应导航数据的150.02位,因此至少要提前跟踪处理 151 位的导航数据才能保证正确的跟踪解调出后面的导航数据。考虑到导航数据信噪比的影响,本文采取了预读 200 位导航数据的策略[11]。

3.5 后续帧组合方法

处理完毕以后的数据要重新组合一下,成为一个完整顺序正确的前三子帧数据,但是在此之前应该考虑到现在这个顺序的三帧数据并不是在同一个页里面,因此应考虑可能造成的影响,先分别分析一下前三帧用于用户位置解算的数据[12]:

在第 1 帧中,除了星期数 TOW 以外别的参数在相邻的两个整页之间可以视为没有变化,而 TOW 的值可以根据相邻子帧的值推算出来,TOW 在某一个子帧里面的值表示的含义是下一子帧发射时,它的单位 1 代表 6s,因此,在计算伪距时以解调出来的第一个子帧的 TOW 值乘以 6,然后再减去 6s,就是当前子帧被发射时刻的时间参数[13,14]。

在第 2 和第 3 子帧中主要包含的是卫星轨道信息,通过对大量数据的前后页的同号子帧进行对比,以及使用相邻页的子帧组合成完整的前 3 帧数据进行定位,得到的结果均与使用同一页中的前 3 帧基本相同,因此,在后续的帧组合过程中需要处理好 TOW 字和计算出准确的伪距。自适应帧选择算法通过容错帧同步方法结合奇偶校验的方式来快速确定接收到的初始帧帧号,然后根据帧号动态处理接下来的数据,从而高效地实现导航电文前 3 帧的提取。

4 算法分析与测试结果

此算法是从原始数据源出发来减少数据量及计算量,因此其效率的提高可以预先粗略地计算出来。在上面的算法程序中,接收机接收到的导航数据的起始点是完全随机的,因此 *BeginIndex* 的取值是 0 到 6000 的平均分布,为了计算算法的平均效率,取其平均值:3000。在 Step3 中,由于对其它的奇偶校验位进行了检测,使得在程序中出现重新检测的几率几乎是零,在大量的实验中未出现重新检测的情况,因此,对于 $FrameID = 1$,处理的总数据量为 $12000 + 6000 + BeginIndex = 21000$;对于 $FrameID = 2$,处理的总数据量为 $12000 + BeginIndex + 200 + 6000 = 21200$;对于 $FrameID = 3$,处理的总数据量为 $12000 + 200 + 6000 \times 2 = 24200$;对于 $FrameID = 4$,处理的总数据量为 $12000 + 200 + 6000 \times 3 = 30200$;对于 $FrameID = 5$,处理的总数据量为 $12000 + 200 + BeginIndex + 6000 \times 2 = 27200$。这 5 种情况出现的概率是一样的,因此平均处理的数据量为 $(21000 + 21200 + 24200 + 30200 + 27200)/5 = 24760$。与先前的数据处理总量 36000 相比,处理效率理论上提高了 31.22%。算法包含一些判断和附加策略,相应的操作会耗掉一小部分时间,本文在对大量数据进行试验后对结果取了平均。另外,对于内存的开销实现了很大程度的削减,为了提高接收机跟踪阶段的效率,往往采用一次性把 6 个子帧的数据读取到内存,这样虽然能有效地减少内存读写次数,但是对于内存的最大容量提出了很高的要求,因为对于 6 个子帧的原始导航数据,其容量约为 200 兆字节,本算法把跟踪阶段分割为两部分处理,首先处理 12000 位数据,接下来再处理后续的数据,可以有效地利用内存的动态分配,这样在损失一小部分时间的条件下大大降低了内存的最大开销值。对于传统算法和自适应选帧算法下整个接收机的总体性能进行了对比测试,并且对两种算法下内存的使用情况

作了跟踪。下面给出实验结果。

图 5 和图 6 是性能效率图示,图 7 和图 8 是资源效率图示。本实验使用了两种数据,一种是仿真数据,采样率为4.096MHz,一种是从RF前端获取的真实导航数据,采样率为 5.714MHz。本实验连续测量了跟踪解调阶段的内存使用情况(图 7 和图 8),自适应选帧算法采取 *BeginIndex* 的均值 3000 作为基准情况进行试验,仿真数据可以人为的确定接收数据的起始端。本实验取导航数据整页的任意

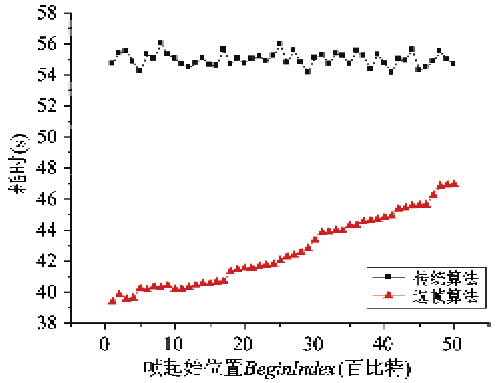

图 5 模拟数据性能对比图

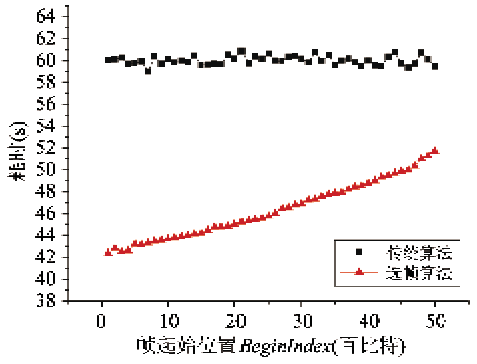

图 6 真实数据性能对比图

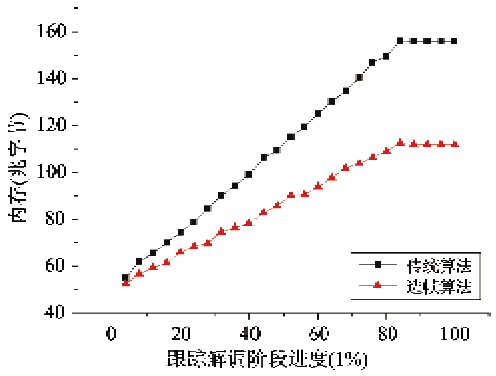

图 7 模拟数据内存分配状态图

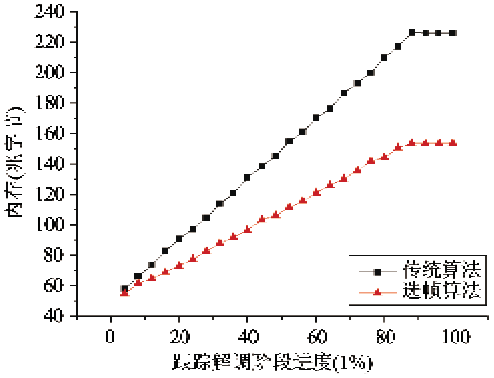

图 8 真实数据内存分配状态图

位置作为起始端来验证此算法,可以看出,当 *BeginIndex* 值较小时选帧算法对性能的提升非常大,而 *BeginIndex* 值对传统算法没有影响,在对大量的真实数据测试后得到的结果与使用仿真数据基本吻合,实际测试的时间开销减少了平均 30.16%,与理论值的差值主要表现在算法判断策略的开销以及增加的内存读取开销上,最大内存开销削减了平均 31.20%。

在定位精度方面,将自适应帧选择算法与传统的处理算法进行了对比。本实验的测试数据是采用 30 组在不同地点不同时间接收到的导航数据,并且接收机所在位置的精确经纬度是已知的,将这些导航数据分别用传统算法跟自适应帧选择算法进行处理,最终解算出经纬度坐标,然后分别计算这 30 组结果与基准位置的欧氏距离误差。从图 9 中可以看出,当在自适应帧选择算法处理过程中没有出现跳帧处理时,计算的结果跟传统算法相同,当出现跳帧情况时,由于前 3 帧数据不在同一个导航页会出现一定的误差,但是这个误差非常小,控制在几米范围内,因此对于精度的影响不是很大。

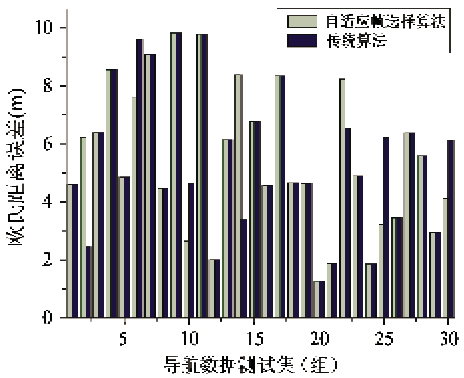

图 9 定位精度对比图

5 结 论

在软件 GPS 接收机中计算量最大、消耗资源最多的就是跟踪阶段,如何提高跟踪阶段的效率、缩短 TTFF 成为软件接收机实现实时化的重要任务。本研究从接收机的实际应用模式出发,以提高接收机初次定位的效率为目的,首先通过预处理一段导航数据来判断初始帧的帧号,然后根据相应的策略有选择地处理接下来的导航数据,舍弃与初始定位无关的帧,从而大大减少了跟踪阶段的数据处理量,提高了初次定位的效率。另外,由于读入数据的减少,也大幅度地减少了内存开销。实验结果验证了算法的可行性,减少了跟踪阶段的处理时间以及资源开销,大幅提高了初次定位效率,为后期的嵌入式平台上的实时化导航奠定了基础。后期的工作主要是搭建软件接收机实时处理平台,验证自适应选帧算法在接收机整体应用上的有效性,以及实时处理环境下的算法改进。

参考文献

[1] Lita L, Visan D A, Popa H. Localization System Based on Enhanced Software GPS Receiver. In: Proceedings of the ISSE 29th International Spring Seminar on Electronics Technology, st. Marienthal, Germany, 2006. 350-354

[2] 李跃. 导航与定位:信息化战争的北斗星. 北京:国防工业出版社,2008. 5-30

[3] Ward P W. Performance comparisons between FLL, PLL and a novel FLL-assisted-PLL carrier tracking loop under RF interfe-rence conditions. In: Proceedings of the 11th International Techni-ca Meeting of the Satellite Division of the Institute of Navigation, Nashville, USA, 1998. 783-795

[4] Roncagliolo P A, De Blasis C E, Muravchik C H. GPS digital tra-cking loops design for high dynamic launching vehicles. In: Proceedings of the 9th International Symposium on Spread Spectrum Techniques and Applications, Manaos, Brazil, 2006. 41-45

[5] 姜毅,张淑芳,胡青等. 一种低复杂度 GPS 载波跟踪环路设计. 电子学报, 2010,38(12):2822-2826

[6] 苗剑峰,孙永荣,陈武等. GPS 软件接收机并行处理的实现. 应用科学学报,2009,27(2):203-209

[7] Tsui J B Y. Fundamentals of Global Positioning System Receivers : Software Approach. 2nd Ed., United States of America: John Wiley & Sons, Inc., 2005. 80-97

[8] 覃新贤. 高灵敏度 GPS 软件接收机关键技术及开发平台研究:[博士学位论文]. 北京:中国科学院计算技术研究所,2009. 44-50

[9] 杨俊, 武奇生. GPS 基本原理及其 Matlab 仿真,西安:西安电子科技大学出版社,2006. 98-120

[10] Psiaki M L. Block acquisition of weak GPS signals in a software receiver. In: Proceedings of the ION GPS 2001, Salt Lake City, USA. 2838-2850

[11] Gardner F M. Phaselock Techniques. 3rd Ed., Hoboken: Wiley-Interscience, 2005. 101-230

[12] 刘基余. GPS 卫星导航定位原理与方法. 北京:科学出版社,2003. 197-207

[13] Akos D. A Software Radio Approach to Global Navigation Satellite System Receiver. Athens OH: Ohio University Press, 1997. 1-5

[14] Akos D M. Real-time GPS software radio receiver. In: Proceedings of the Institute of Navigation National Technical Meeting, Long Beach, USA, 2001. 809-816

An adaptive frame selection algorithm for the tracking period of a software GPS receiver

Yao Xiangzhen[*,**], Cui Shaolong[*,**], Fang Jinyun[*]

([*]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]Graduate University of Chinese Academy of Sciences, Beijing 100049)

Abstract

With the purpose of improving the initial positioning efficiency of a software GPS receiver, this paper presents an adaptive frame selection algorithm for its tracking period from the angle of practical application. When using the algorithm, an initial section of the navigation data can be preprocessed to determine the ID of frames, and then an appropriate strategy can be formulated to handle the next navigation data. The frames that are not involved in the initial positioning are dropped, therefore, the navigation data that will be processed is reduced significantly. Finally, the initial positioning efficiency is improved to reduce the time to first fix (TTFF). Because this method starts from the original data sources in the tracking loop to reduce the amount of data and computing, it can greatly reduce the TTFF. The experimental results demonstrate this algorithm can improve the initial positioning efficiency and reduce tracking operations significantly. Most importantly, it provides a solid foundation for the real-time embedded platform.

Key words: software GPS receiver, initial positioning, signal tracking, adaptive frame selection