Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/TX-06/0-4480-1		

4. Title and Subtitle	5. Report Date
Collecting Stop and Go Inertial Profile Measurements	May, 2006
	6. Performing Organization Code
7. Authors	8. Performing Organization Report No
Roger S. Walker, Ph.D. and Eric Becker, MSCSE	Report 0-4480-1
9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
The University of Texas at Arlington 416 Yates Computer Science & Engineering Department Arlington, TX 76019	11. Contract or Grant No. Project No. 0-4480
12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Texas Department of Transportation Research and Technology Implementation Office P.O .Box 5080 Austin, TX 78763-5080	Technical Report, September 2002-August 2004 14. Sponsoring Agency Code

15. Supplementary Notes

Research performed in cooperation with the Texas Department of Transportation and the Federal Highway Administration.

16. Abstract

The Texas Department of Transportation (TxDOT) annually collects profile data over the state-maintained highway network. TxDOT uses the profile data to determine ride quality based on the Present Serviceability Index (PSI). The indices determined are stored in the Pavement Management Information System (PMIS) database, and are published in the PMIS reports prepared annually by the Materials and Pavements Section of the Construction Division. Inertial Reference Profilers, such as TxDOT's Profilers, are adversely affected by stop and go operations, common during PMIS data collections. When the profiler slows below 12 mph or comes to a complete stop, profile measurements are not valid. This situation causes the loss of ride data at intersections, in high traffic conditions, and in urban areas. This report describes a method that TxDOT can uses that will minimize the effects of stop and go for the inertial profile measurements.

17. Key Words	18. Distribution Statement
Present Serviceability Index (PSI), Ride, Inertial Profiler	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161, www.ntis.gov.

19. Security Classif.(of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	52	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Collecting Stop and Go Inertial Profile Measurements

TECHNICAL REPORT 0-4480-1
PROJECT NUMBER 0-4480

PROJECT TITLE: Develop a Methodology for Collecting Stop and Go Inertial Profile Measurements

THE UNIVERSITY OF TEXAS AT ARLINGTON
TRANSPORTATION INSTRUMENTATION LABORATORY

Roger S. Walker, Ph.D., P.E
Eric Becker, MSCSE

Date: May 2006

Notice – The United States Government and the State of Texas do not endorse products or manufacturers. Trade or manufacturers' names appear solely because they are considered essential to the object of the report.

IMPLEMENTATION STATEMENT

During this project a method has been developed that minimizes the effects of stop and go operations during profile data collection. The method developed provides a significant improvement over current operations. WinTK, the profiler data server software, has been modified so the method can be implemented. This new software version is ready for implementation and PMIS field usage.

DISCLAIMER(S)

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official view or policies of the Texas Department of Transportation or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

There was no invention or discovery conceived or first actually reduced to practice in the course of or under this contract, including any art, method, process, machine, manufacture, design or composition of matter, or any new useful improvement thereof, or any variety of plant, which is or may be patentable under patent laws of the United States of America or any foreign country.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Carl Bertrand (retired), the Project Director, and Brian Michalk of the Texas Department of Transportation. Acknowledgements are also due the staff members of the Texas Department and at the Transportation Instrumentation Laboratory facility at the University of Texas at Arlington for assistance during the project.

Table of Contents

List of Figures

List of Tables

List of Equations

CHAPTER 1
INTRODUCTION

The Texas Department of Transportation (TxDOT) annually collects profile data over the state-maintained highway network. TxDOT uses the profile data to determine ride quality based on the Present Serviceability Index (PSI). The indices determined are stored in the Pavement Management Information System (PMIS) database, and are published in the PMIS reports prepared annually by the Materials and Pavements Section of the Construction Division. Inertial Reference Profilers, such as TxDOT's Profilers, are adversely affected by stop and go operations, common during PMIS data collections. When the profiler slows below 12 mph or comes to a complete stop, profile measurements are not valid. This situation causes the loss of ride data at intersections, in high traffic conditions, and in urban areas. This research will develop a methodology that will minimize the effects of stop and go for the inertial profile measurements.

Background

Introduction:

There are a number of factors that affect profile measurements using inertial reference profilers at slow speeds. For example, one major factor is the measurements from the accelerometer used for vehicle body vertical accelerations. The accelerometer provides a voltage proportional to the body acceleration. This voltage is then digitized and used along with the road-body displacements as measured by a laser to obtain an estimate of the road profile that the profiler vehicle is traveling. The accelerometer readings provide a measurement of the second derivative of the vehicle body with respect to time (acceleration). As the profiler vehicle slows, the accelerations and associated voltages become very small affecting the measurements. If small but abrupt changes occur, such as the vehicle being stopped and then started forward a vertical impulse component can occur. The voltage component can be significant when compared to the previous readings. The effect of this in the road profile computations results in wide swings in the computed profile. The effects of Stop and Go (SAG) operations on profile from one of the TxDOT profilers are illustrated in Figure 1. For this figure, the profiler is initially traveling at a speed of about 40 MPH before slowing to a complete stop, and then starting back up to the original speed.

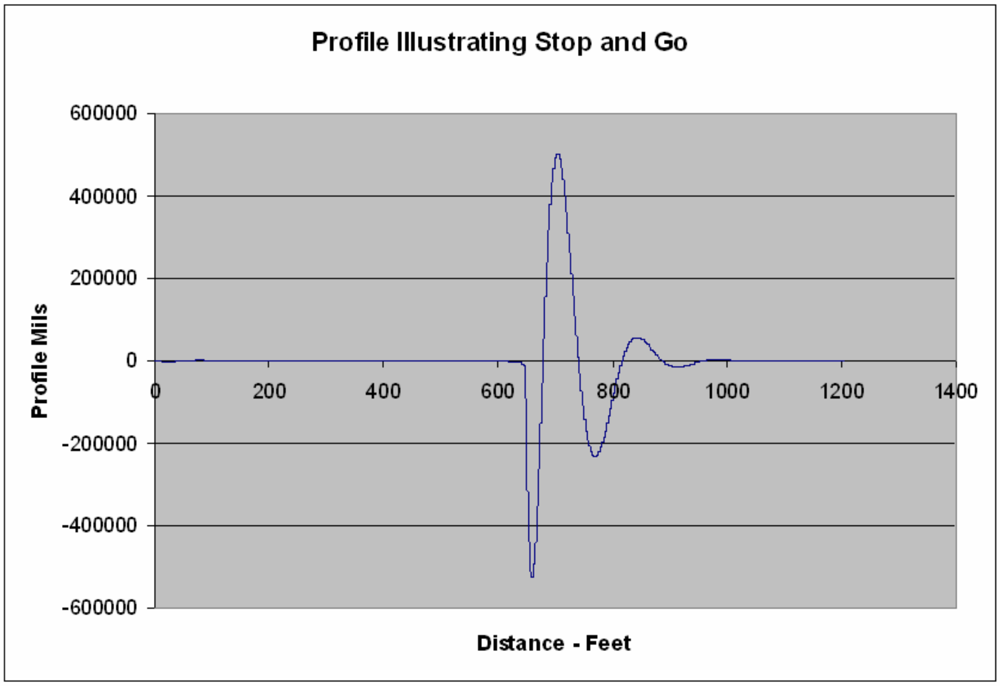

Figure 1.1 Efforts of Stop and Go On Profile From TxDOT Profilers

As might be expected, the effects initial start of profile measurements are similarly distorted as SAG. Filtering considerations need to be accounted for during bounce tests and other profile sensor testing. Figure 1.2 illustrates the startup effects on a typical profile data run. The initial profile data has a similar amplitude swing as that of SAG. This similarity is because of the type and wavelength of the filter used. The shape of this filter is directly related to the wavelength of the filter used for profile computation.

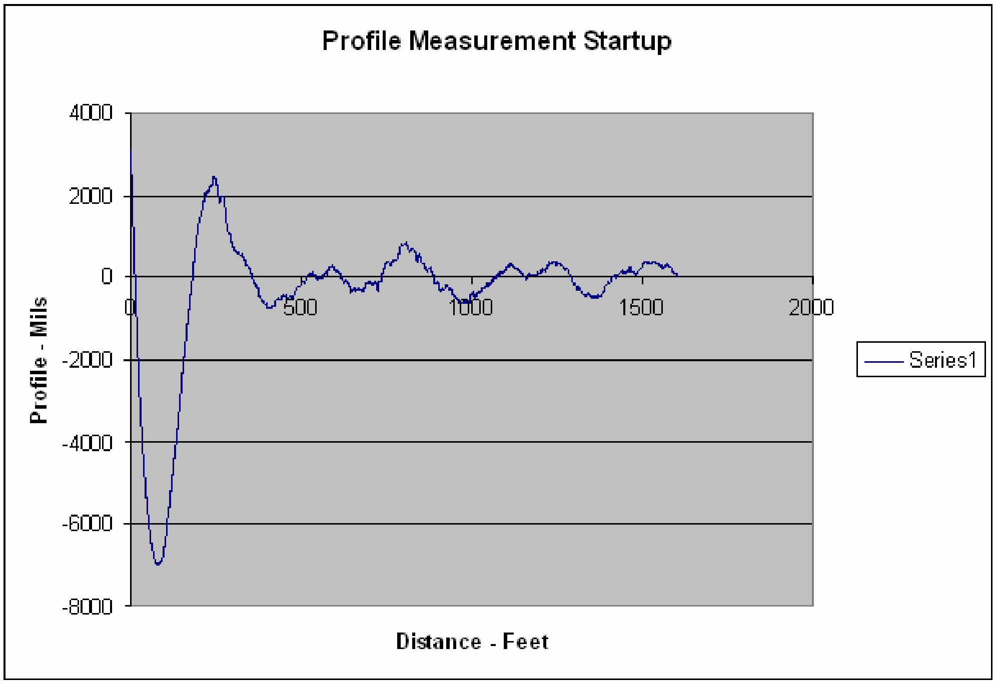

Figure 1.2 Efforts of Initial Startup On Profile of TxDOT Profilers

Factors Affecting Profile Measurements:

Modern inertial reference profilers use high pass digital filters to reduce the effects of the long wavelengths on profile computations. Digital filters are categorized in terms of their impulse responses and are either finite (FIR filters) or infinite (IIR filters) in duration. The use of one type over the other is typically determined by the application requirements. For example, IIR filters have very sharp frequency cutoff characteristics that can be attained with low-order structures (fewer coefficients). FIR filters take many more coefficients for similar sharp cutoff characteristics. The length required for the correct start up response is affected by the filter characteristics.

As the amplitudes of the various profile frequencies are attenuated, the phase of each attenuated frequency is changed. The IIR filter has disadvantages in its non-linear phase characteristics. FIR filters are typically used if linear phase characteristics are desired, as they can be designed so that the phase changes are linear. Linear phase results in a constant delay for the attenuated frequencies. Depending on what filtering method is used, adjustments for SAG operations must work with the characteristics of the filter used in order to minimize the effects of errors on profile computation. FIR filters are implemented by convolving the inputs from the signal that is to be filtered with the filter coefficients, IIR filters are implemented using both the filter inputs and previous filter outputs. The amount of settling time or time before the startup effects of the filter is closely related to the wavelength of the high pass filter used in the profile computation. This can be noted in Figure 1.2

Proper mounting is also important for accelerometer usage in profile computations. The accelerometer needs to provide an accurate measure of the body accelerations at the point where the relative displacement between the vehicle and the pavement is measured. The locations in the vehicle can significantly change the accelerations measured by the sensor. Figure 1.3 illustrates accelerations measured from three similar accelerometers at three different mounting locations in a van. The van was traveling at approximately 50 miles per hour and the readings were each sampled at 1 KHz. As can be noted there are significant variations between all three. Since profile is computed from both laser and accelerometer measurements, the appropriate acceleration readings must be used.

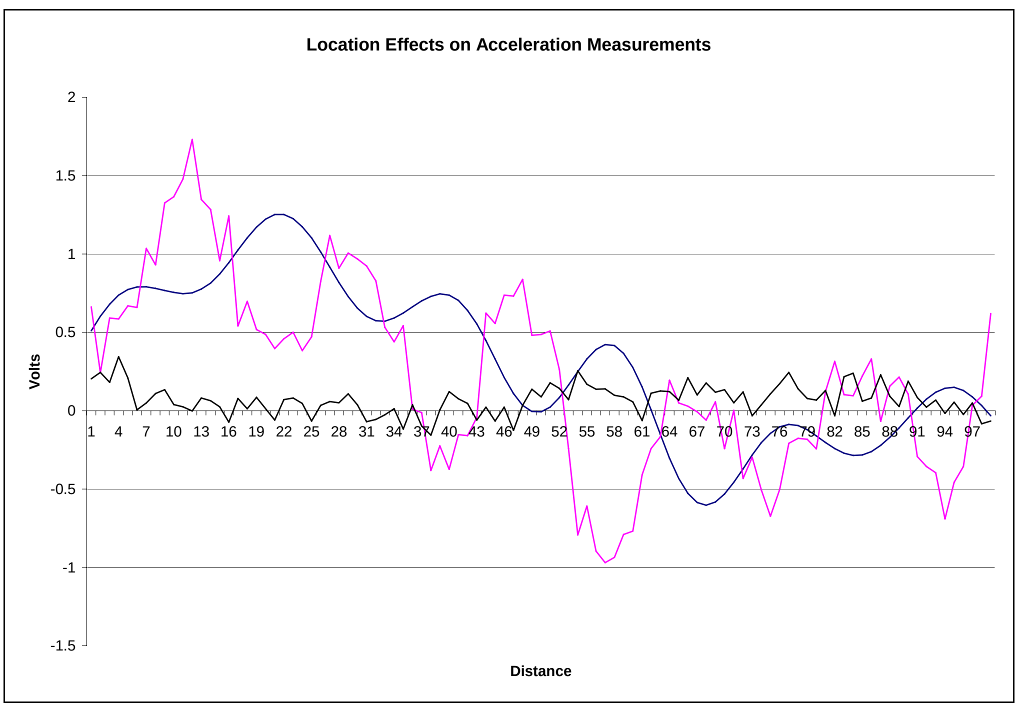

Figure 1.3 The Effects of Different Accelerometer Mounting Locations

At the time of the project, Dynatest was the only company that researchers were aware of that offered profilers with a capability of minimizing the effects of SAG operations on inertial reference profilers, although their systems do not meet the same accuracy and repeatability as the continuous higher speed measurements. The procedures used by Dynatest for profile computation, as in the case of other profiler manufacturers, are not publicly available.

Report Contents:

The following chapters will discuss the methods used to minimize the effects of SAG on data collection with the TxDOT Profilers, including the data collected and corresponding analysis. Since it was not desired to change the profile computation procedures used for high speed measurements, efforts are focused on adjustments that can be made to the profile module that will minimize the effects of stop and go operations on the current profilers. Chapters II and III describe the methods applied to minimize the effects of stop and go operations on road profile data and the resulting statistics, IRI, SI, and NSI. Chapters II and III include the final method applied and discusses the necessary changes made to WNTK.

Chapter IV provides results of using the final procedure and Chapter V provides a summary of the research findings and recommended procedures for data collection with the TxDOT Profilers.

CHAPTER 2
METHODS USED TO MINIMIZE EFFECTS OF SAG

Methods considered for minimizing the effects of SAG operations on the TxDOT profilers are discussed in this chapter. When first proposing the project it was thought that corrections to profile could easily be accomplished by simply filtering the effects of the sensors or the computed profile when operating speeds dropped below 10 MPH. After a number of initial attempts in adjusting the filter parameters, it became apparent that the process would require more effort than originally thought. As will be illustrated the final approach involved combinations of both attenuation of profile data and variations in the sampling rate reported to the profiling module thus affecting the filtering process in the profile computation module during profile measurements. Filtering efforts will first be described.

Filtering

As noted it was initially thought that varying the cutoff frequencies of the filters used for computing profile would provide a quick and somewhat easy fix.[1] The cutoff frequency is selectable by the operator in the profiler's configuration dictionary[2] this could be done by simply changing the cutoff frequency in accordance with the profiler speed. The profiler object module has the provision for applying an attenuation factor to each of the sensor signals used in computation by including set commands for each of the sensors. These options were initially tried, but without much success, and thus the results of these efforts will not be addressed. Instead the report will focus on the procedures used that involved the combinations of the methods which help reduce the effect of the filter and provide the

---

[1] If there was no filtering required, then there wouldn't be a SAG problem as the wide swings in the profile data is because of the filtering action.

[2]The configuration dictionary is the list of parameters used by client programs for initiating, operating, and controlling the WinTK server program during testing and data collection.

best results. Since large variations in profile occur during the SAG process, the first method discussed will be to minimize or attenuate the profile during the SAG operations

Profile Attenuation

The profile amplitude variation was the most obvious problem that affected the profile when the vehicle slowed to a stop and then started up from a stopped position. As discussed in Chapter One, during initial start up wide swings in the profile exist because of the high pass filtering used to attenuate the low frequencies or long wave lengths. Figure 1.2 illustrates the profile during initial start up operations. As was noted, the profile is significantly affected by the filter before settling down. The distance required for settling varies but is related to the filter type and cutoff frequency.. Since filtering is required for inertial reference profilers using accelerometer based systems, methods were focused on reducing the undesired side effects of this filtering. Either the filter characteristics need to be changed or attenuation applied to the profile at these times to prevent the wide swings. The approach used was to reduce this effect yet still provide reasonable profile estimates.

Profile attenuation can be applied in several places in profile data collection. It can be applied either to the individual sensor readings before transferring to the profile module, using one of the module set commands to perform in the module, or to the computed profile. Attenuation can also be applied in the profile module during filtering.

The Three Problems: When, Where and How Much?

To address attenuation methods, the Stop and Go question for the profiler had three main issues that needed to be addressed. The first was how to minimize the effect of the filter swing when the vehicle starts to slow down. The second issue was to identify the actual speed when the SAG problem occurs. Finally, at what speed or time does the profile computation procedure recover from the SAG effects?

The Speed Problem

As discussed above the first issue that needed to be addressed was to identify at what point does the profile began to be adversely affected by vehicle speed. A ‘SAG Window’ will be defined as the portion of time and thus distance where the profile is adversely affected by the SAG actions and which corrective action is to be applied. The profiler uses a state machine that recognizes a distance signal, which is used by the system to indicate that the vehicle has moved a specified distance. A rectangular wave form or pulse is used to distinguish between the distance points. The time between pulses is related to vehicle speed. Various variables are used in the state machine during the real time sensor data acquisition. The sensor signals used in computing profile are sampled with respect to time and reported for each distance interval. The Time Factor (TF) is one of the state machine variables and is used to indicate the number (or frequency) of time samples obtained over a specific distance interval.[3] The time factor is the measure of the number of raw sensor readings received per each distance signal. Thus the filter time factor is proportional to the inverse of the speed, and increases as the vehicle approaches a speed of zero. This value was observed and recorded and used to indicate where in the time of a data collection that the SAG Window begins. It could then be used to solve the other two issues. Figure 2.1 illustrates a typical relationship observed for the Time Factor as a function of speed. As may be observed from this figure, the faster the speed, the smaller the TF becomes.

The Amplitude and Frequency Problems

The second issue is a direct consequence of the first, and that is, when the data collection vehicle comes to a stop, the sampling frequency and other parameters sent to profile module result in its output increasing its value by several orders of magnitude. The question now was how to correct the amplitude over the period that was affected by the SAG, and still have the system function continuously afterward. In the end, three areas were considered in answering the amplitude and frequency problems. The three

[3] The profiling method used in the profile module is a modified form of the South Dakota process which samples and integrates the laser and acceleration signals as a function of time.

procedures investigated to minimize these effects were the usage of an attenuation factor, sensor replacement methods, and the Time Factor.

The first method applied was a variable attenuation factor.

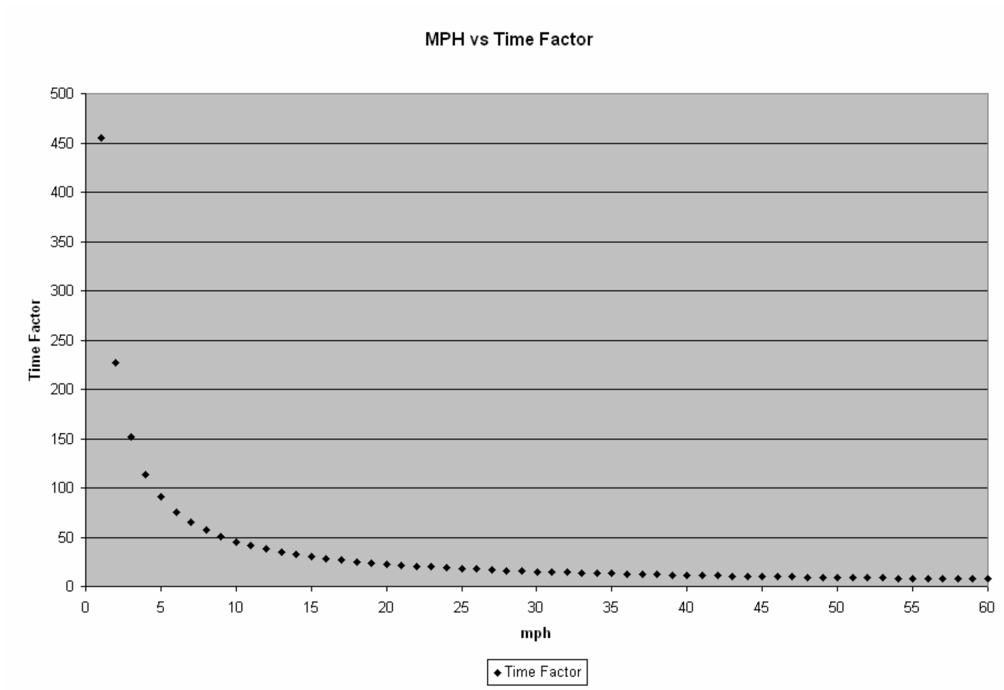

Figure 2.1 Speed vs. Time Factor

The attenuation factor variable was used to attenuate the computed profile and thus reduce the scale of the output when the vehicle slowed to a stop and then started moving again. The idea was to shrink the amplitude by applying the variable to each output until the vehicle had started moving again. This approach did not behave as expected, and was rejected because the amplitude was not within an acceptable range.

The second approach was to try replacing the data collected during the SAG area with more acceptable readings to help minimize the effect of the actual data. Rather than using the actual profile data, methods of obtaining data from another source were tried.

Value Replacements

The first replacement method tried was to replace the actual profile in the disrupted section, or SAG Window, with a known good sample of profile data. This idea could work, but was rejected. While it would handle the question of amplitude and location, the data in the region would not be accurate. A second idea of replacing all the affected points with the last known acceptable profile was considered, but was rejected for the same reasons.   An example of both of the methods is illustrated in Figure 2.2.

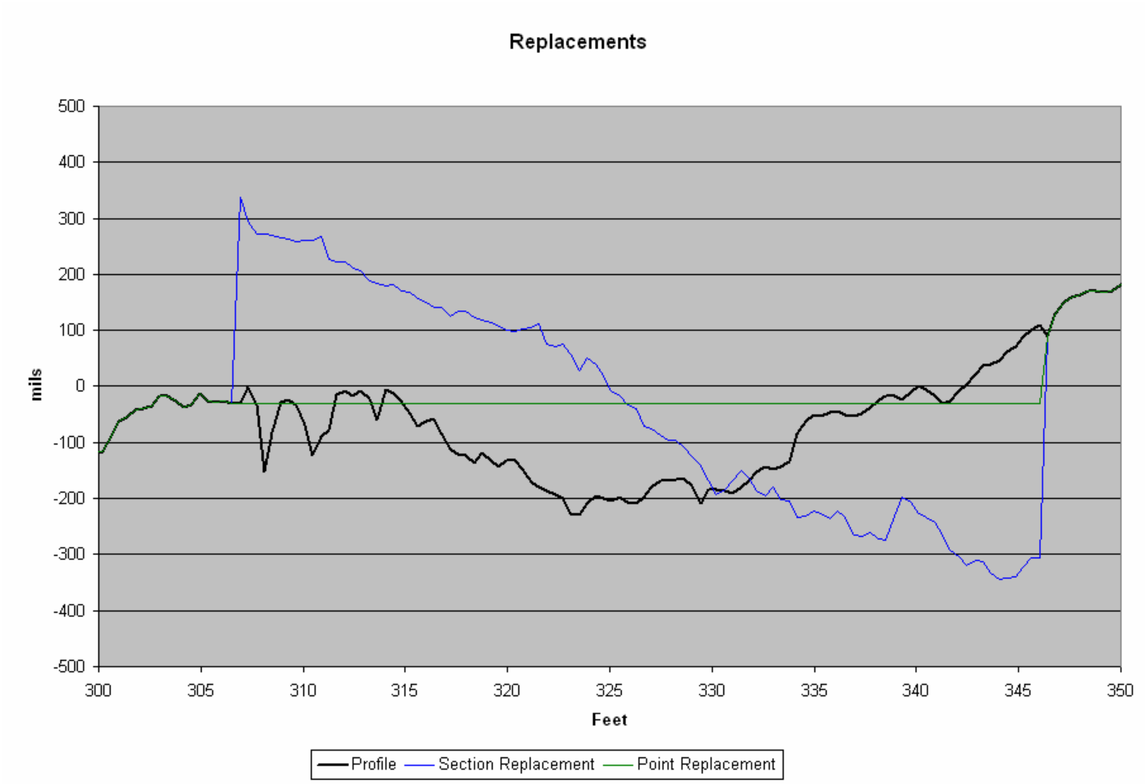

Figure 2.2 An Example of Using Section and Point Replacements

Formula Replacements

Another method tried was to provide or replace the data set acquired by the sensors with a different set of sensor inputs. For this replacement, two methods were tried. The first method was to substitute the laser readings used for computing the profile during the SAG Window with a computed substitute, trying to bring down the amplitude to acceptable ranges. For this method, the profile was replaced in the SAG Window by a new value which was a function of the original computed profile Pc, the laser reading, L and an alpha value, α and where α is bounded by zero and one, or,

$$P=\alpha*P+(1-\alpha)*L$$

The results were not good, since the laser readings did not produce the desired effect of reducing the amplitude. The use of this method is illustrated in Figure 2.3 for starting from a stopped position and for a SAG operation.

Figure 2.3 Alpha Replacement Method

Timer Replacement

An additional method for handling the SAG is to use the clock and timer mode of the profiler in order to keep the amplitude in the correct range. For this case, distance pulses are sent to the profiler in accordance with a separate time signal. The result would be that the profile would always be within the correct range, but the location of the data would shift, that is in the time mode, the distance is simulated by a fixed distance. For example, profile would be computed as if the vehicle was actually moving at a constant speed but the vehicle would actually be slowing or sitting still. In this case it was noted that the amplitude problem would be acceptable, but the distance would have been increased by many additional data points. An example of this is illustrated in Figure 2.4 below.

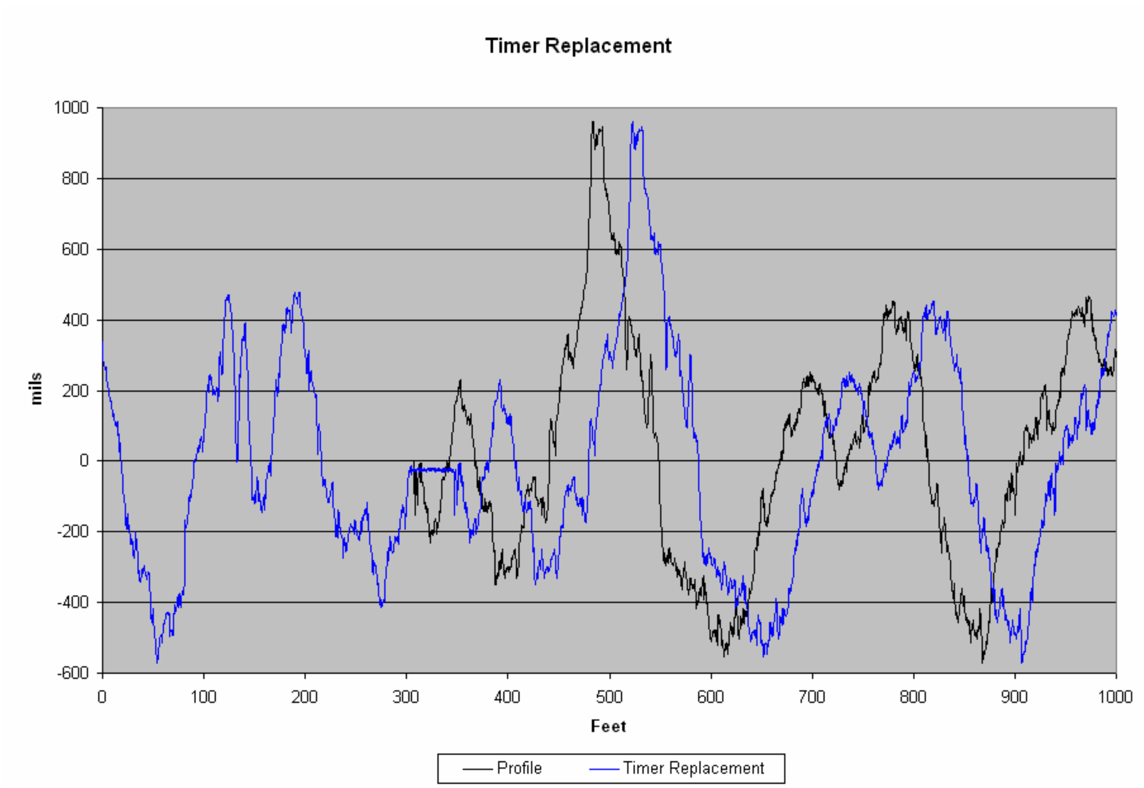

Figure 2.4 Timer Replacement

Time Factor

In the previous sections, it is noted that in tests and methods to control the SAG condition, no single routine seemed to be the best case for handling all the questions of amplitude, speed, and frequency. When simulating a fixed speed by using a constant time pulse in lieu of a distance pulse, the wide swings in the profile would not occur, but of course the distance traveled would be incorrect. The other replacement methods had correct distances, but had unusable amplitude.

As noted when using the timer replacement, the system was on a timer instead of the distance signal and the time factor value, TF or number of time samples per distance signal, remained a constant throughout the routine. It was then decided that maybe a procedure of involving TF itself could solve the problem. The first tests with just the Time Factor by itself were promising. The values of the amplitude shrank nearly to the acceptable range, without shifting the distance of the signal because of the difference between the actual speed and the simulated speed. Then, researchers noticed that if the SAG window was expanded to include the particular place where the vehicle decelerated, and then later on when the vehicle accelerated instead of at just the point of where the vehicle stopped, the results became closer still.

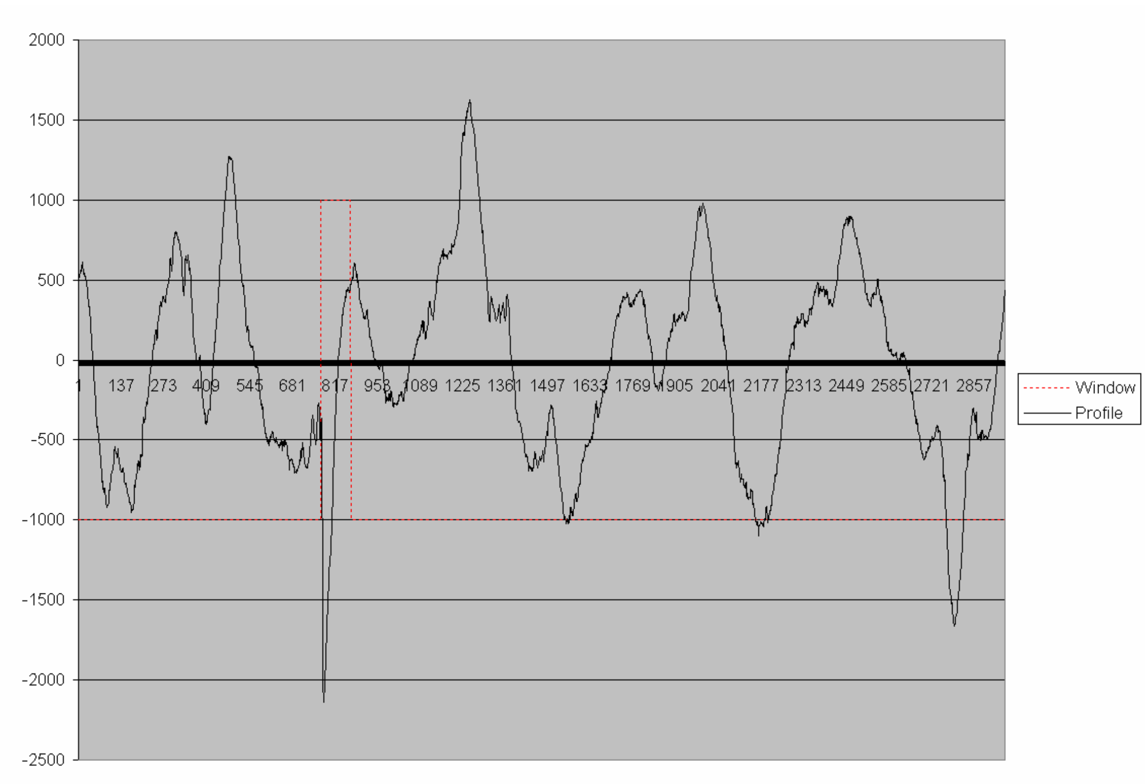

Figure 2.5 Profile with Window beginning at the Stop

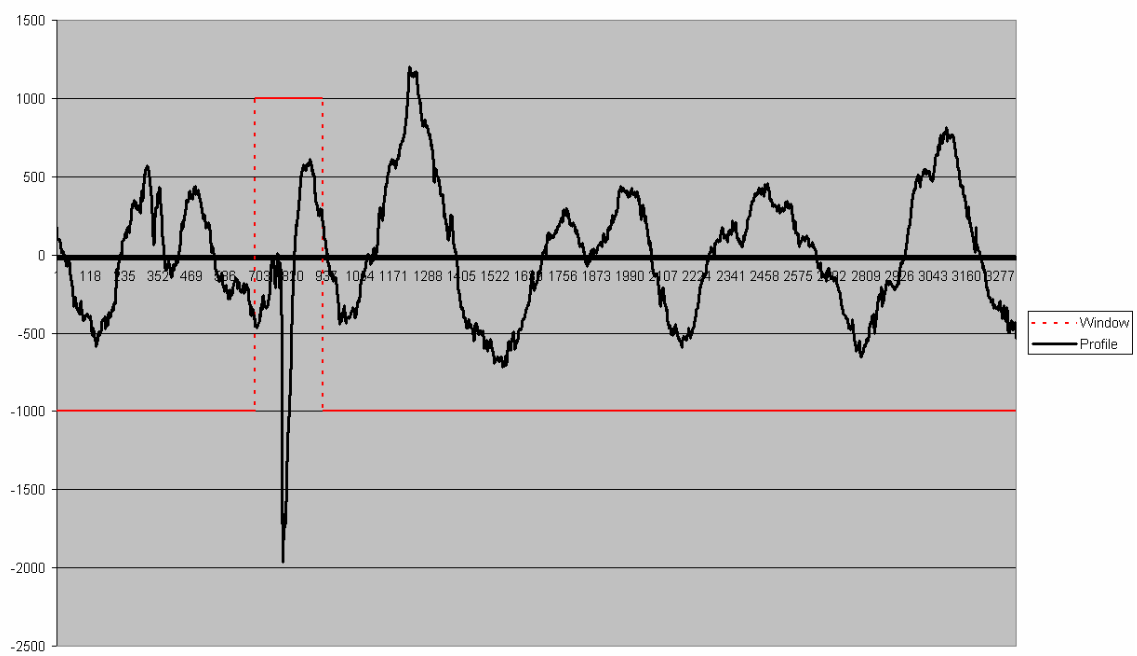

Figure 2.6 Profile with Expanded Window

This time and distance approach on its own was nearly acceptable. Swings in profile would occur, but the effect was much less dramatic. To control the range values of the output of the profile method, a slew was added. A slew routine was placed upon the output to help prevent rapid rises in the amplitude. The match became even closer at this point, as can be seen in figure 2.7. One final method was tried, shown in figure 2.8, adding an averaging on the last few outputs of the profile in the SAG window, but this addition did not improve the result, so it was removed.

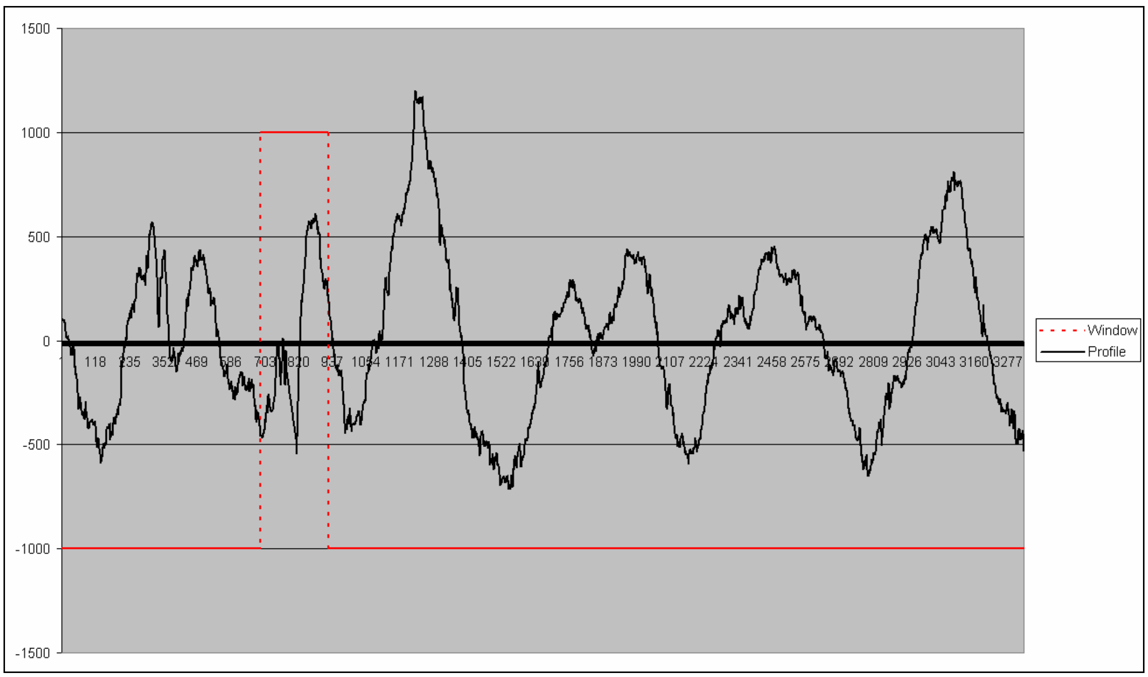

Figure 2.7 Profile with Expanded Window and Slew

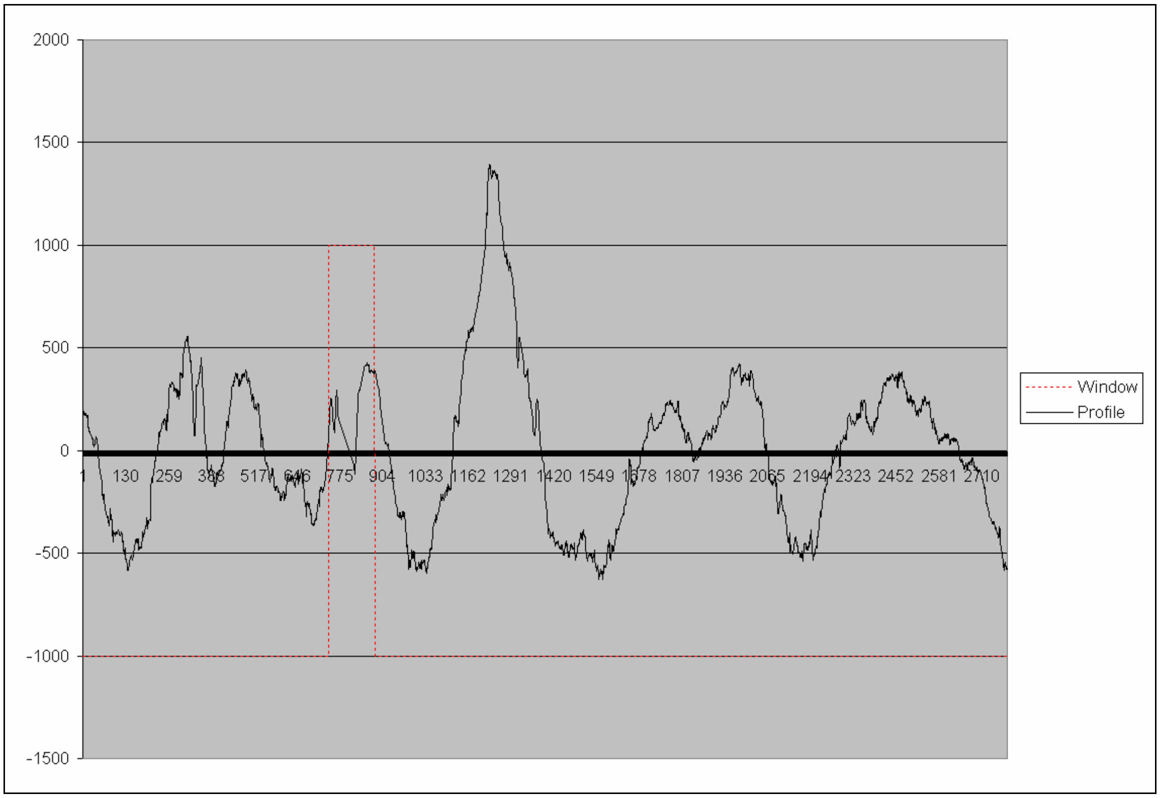

Figure 2.8 Profile with an expanded Window and Slew and Averaging

As discussed, after trying the various variations in the parameters and variables applied to this module, the result that best reduced the wide variations, yet did not dramatically affect the profile values and summary statistics, was altering the Time Factor within the SAG expanded window and the use of a slew routine. This result was the best match for the desired location, and amplitude of the output of the profile module. In the next chapter the integration of the method with WinTK server is discussed.

CHAPTER 3
Integration of SAG Algorithm With WinTK Server Code

In this chapter, the procedures used to implement a SAG function that minimized the effect of SAG operations within WinTK are addressed. The implementation will be performed in two phases or stages. First, however, the speed in which the SAG function is enabled must be converted to the appropriate units for the SAG routine. The first phase of the routine will then be to calculate a new time factor[4] of FT from the old FT value. Then in Phase 2, the slew routine would be applied.

Converting Speed to Time Factor

One of the requirements of the SAG routine is to be able to detect the window of velocities in which the routine is active. From a user perspective, the desired speed is in miles per hour, while the state machine is expecting to see a Time Factor.

Given a desired speed, either for the Stop Speed or the Go Speed, the sampling rate of the distance channel on the A/D converter board, and the distance between distance pulses, a limiting number of samples per distance signal can be calculated. This may be done before the state machine starts and be used to compare with the state machine count using the following speed conversion procedure.

$$\left(\frac{\text{Sampling Rate(Hz)}}{\text{Speed (kps) SpeedCount } * \text{ Distance Counts per Kilometer}}\right) = \text{Samples per Distance Interval}$$

Equation 3.1 Calculating the Time Factor

[4] The variable `mw_f2` is used for the Time Factor (FT) in the software. The variable m_dSGSpeedCount is used for indicating samples per unit distance, and the variable m_dSGSampleRate is used for the sampling rate.

For example, suppose the SAG window has a ‘Start’ setting in the SAG window of 18 MPH. Also assume that there is a distance pulse every 4 inches and the sampling rate is fixed at 2000 Hz (the default value). Then the profiler computation routine would switch on the Stop and Go routine when the vehicle speed is below 18 mph. A single integer value can be determined as:

$$\left(\frac{1}{18\text{ mph}}\right)(2000\text{ Hz})(4\text{ inches/Distance Interval}) =$$

Or, 25.5 samples per distance interval.[5]

Therefore, the limit for the count for 18 mph is 26 samples over a single distance interval. The Stop and Go speed is changed into an integer limit during the collection setup, and then quickly compared to the Time Factor by the code in Figure 3.1 and activated by the code in Figure 3.2

.

```
m_dSGSpeedCount   = m_pKCFG  ->SpdCnt;
m_dSGSampleRate   = m_pKCFG  ->SamplingRate;

m_dSGSpeedStop    = m_pSAGCFG->m_dSpeedStop*1.609/3600.0;
m_dSGSpeedGo      = m_pSAGCFG->m_dSpeedGo  *1.609/3600.0;

m_iSGF2LimitStop  =
      (int)(m_dSGSampleRate/m_dSGSpeedStop/m_dSGSpeedCount);
m_iSGF2LimitGo    =
      (int)(m_dSGSampleRate/m_dSGSpeedGo/m_dSGSpeedCount);
```

Figure 3.1 Code for finding the Stop Speed and the Go limits.

5

$$\left(\left(\frac{1}{18\text{ mph}}\right)\left(\frac{1\text{ mile}}{5280\text{ feet}}\right)\left(\frac{3600\text{ sec}}{1\text{ hour}}\right)\right)\left(\frac{2000\text{ samples}}{1\text{ sec}}\right)\left(\left(\frac{4\text{ inches}}{1\text{ Distance Interval}}\right)\left(\frac{1\text{ foot}}{12\text{ inches}}\right)\right) =$$

$$\left(\frac{1\text{ sec}}{26.4\text{ feet}}\right)\left(\frac{2000\text{ samples}}{1\text{ sec}}\right)\left(\frac{0.333\text{ foot}}{1\text{ Distance Interval}}\right) = \frac{25.25\text{ samples}}{1\text{ Distance Interval}}$$

```
//-------------------------------------------------------------
//---Stop and Go Detect Stop and Go Range
//-------------------------------------------------------------
if (m_bSGF2Toggle==FALSE)
{
  if (mw_f2>m_iSGF2LimitStop)
  {
    m_bSGF2Toggle=TRUE;
  }
}
else
{
  if (mw_f2<m_iSGF2LimitGo)
  {
    m_bSGF2Toggle=FALSE;
  }
}
```

Figure 3.2 Code for Stop and Go Activation

Once the SAG operation is activated, the SAG operation is then implemented using two phases, the first altering the Time Factor or TF, and the second applying a slew to the computed profile value.

Phase 1: Altering the Time Factor

The first phase is to implement the time factor such that it smoothly transforms from the Speed Stop value, where the SAG window begins, and shifts to the new value where the SAG window ends so long as the ratio of the change does not go below 0.1.

$$\text{Let } R = 1 - 10^{-\left(\frac{\text{Speed Go}}{\text{F2}}\right)}$$

$$\lim_{R \to 0.1} \text{F2} = (\text{Speed Go})(R)$$

Equation 3.2 Formula for altering the Time Factor

```
m_iF2=mw_f2;
if (m_bSGF2Toggle)
{
  m_dSGDamping=(double)((1.0*m_iSGF2LimitGo)/(mw_f2*1.0));
  m_dSGRatio=1.0-pow(10,-m_dSGDamping);
  if (m_dSGRatio<0.1)
  {
  m_dSGRatio=0.1;
  }
mw_f2=(int)(m_dSGRatio*m_iSGF2LimitGo);
}
```

Figure 3.3 Code to alter the Time Factor

Phase 2: The Slew

Though the amplitude was now not expanding beyond a usable range of data, the output from the profile module was still changing too rapidly.

```
lAbsDifference=abs(lProfile - *lProfileLast);

if (lAbsDifference>lDifference)
{
  if (lProfile>*lProfileLast)
  {
    iSign= 1;
  }
  else
  {
    iSign=-1;
  }
  lResult=*lProfileLast+iSign*lCorrection;
}
else
{
  lResult=lProfile;
}
```

Figure 3.4 Code for the slew

Integrating SAG

The changes of the SAG solution have to be worked into the running system code that handles the calculation of profile in real time for the WTKServer program. The following flowchart and code show how the SAG routines hook into the system.

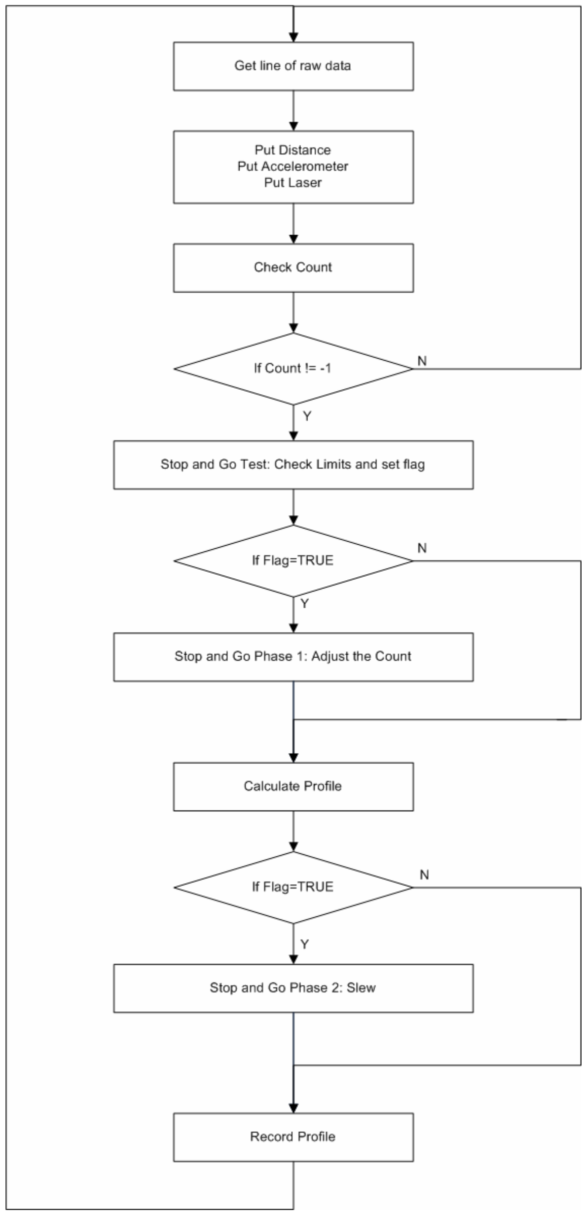

Figure 3.5 Flow Diagram for Initiating SAG During Profile Computations

In the following code, the specific functions illustrated in the flow chart of Figure 3.3 are denoted by a box surrounding the corresponding source code.

```
long ProfileLeft=0;
long ProfileLeftDLL=0;
long ProfileRight=0;
long ProfileRightDLL=0;
BYTE *dataptr = NULL;
unsigned long bufsize=0L;
UINT size=0L;
ECODE status=OLNOERROR;
unsigned int loop=0;
int walk=0;
int bytesize=8;
int iSGCount;
long lData=0;

status = olDmGetValidSamples( hBuf, &bufsize );
status = olDmGetDataWidth( hBuf, &size );
status = olDmGetBufferPtr( hBuf, (void **) &dataptr);

/** KNOWN TO WORK-DO NOT LOSE->KEEP AS REFERENCE
for (loop=0;loop<(bufsize*size);loop+=size*m_iNumberOfChannels)
{
long pathlength=size*m_iNumberOfChannels;
for(walk=0;walk<pathlength;walk+=2)
{
lData=(long)dataptr[loop + walk]+(long)dataptr[loop+1 + walk]*256;
fprintf(m_pFile,"%d\t",lData);
}
fprintf(m_pFile,"\n");
}
/**/
//====Fetch Data=====================================================

for (loop=0;loop<(bufsize*size);loop+=size*m_iNumberOfChannels)
{//BEGIN MAIN FOR LOOP
  long pathlength=size*m_iNumberOfChannels;
```

GET LINE OF RAW DATA.

```
    //----Distance and Trigger--------------------------------------------
    /*C 0*/ m_lColDistance  =(long)dataptr[loop + 0]+(long)dataptr[loop+1 + 0]*256;
    /*C 1*/ m_lColTrigger   =(long)dataptr[loop + 2]+(long)dataptr[loop+1 + 2]*256;
    //----Profile---------------------------------------------------------
    /*C 2*/ m_lColLaserLeft  =(long)dataptr[loop + 4]+(long)dataptr[loop+1 + 4]*256;
    /*C 3*/ m_lColLaserRight =(long)dataptr[loop + 6]+(long)dataptr[loop+1 + 6]*256;
    /*C 4*/ m_lColAccelLeft  =(long)dataptr[loop + 8]+(long)dataptr[loop+1 + 8]*256;
    /*C 5*/ m_lColAccelRight =(long)dataptr[loop +10]+(long)dataptr[loop+1 +10]*256;
    //----Laser Rut and Gyroscope-----------------------------------------
    if((!m_bLaserRut)&&(!m_bGyroscope))
    {
      /*   */ //NO-OP This Space Intentionally Left Blank
    }
    if((!m_bLaserRut)&&( m_bGyroscope))
    {
      /*C 9*/  m_lColGyroX =(long)dataptr[loop +12]+(long)dataptr[loop+1 +12]*256;
      /*C10*/  m_lColGyroY =(long)dataptr[loop +14]+(long)dataptr[loop+1 +14]*256;
    }
```

```
    if(( m_bLaserRut)&&(!m_bGyroscope))
    {
      /*C 6*/ m_lColLaserLeftWing  =(long)dataptr[loop +12]+(long)dataptr[loop+1
+12]*256;
      /*C 7*/ m_lColLaserRightWing =(long)dataptr[loop +14]+(long)dataptr[loop+1
+14]*256;
      /*C 8*/  m_lColLaserCenter    =(long)dataptr[loop +16]+(long)dataptr[loop+1
+16]*256;
    }

    if(( m_bLaserRut)&&( m_bGyroscope))
    {
      /*C 6*/ m_lColLaserLeftWing  =(long)dataptr[loop +12]+(long)dataptr[loop+1
+12]*256;
      /*C 7*/ m_lColLaserRightWing =(long)dataptr[loop +14]+(long)dataptr[loop+1
+14]*256;
      /*C 8*/  m_lColLaserCenter    =(long)dataptr[loop +16]+(long)dataptr[loop+1
+16]*256;
      /*C 9*/  m_lColGyroX =(long)dataptr[loop +18]+(long)dataptr[loop+1 +18]*256;
      /*C10*/  m_lColGyroY =(long)dataptr[loop +20]+(long)dataptr[loop+1 +20]*256;
    }


//*****************************************************************************************
************
    //   Check the Start Signal

//*****************************************************************************************
************
    if(!mw_iStartFlag)
      if(m_lColTrigger<40960)
      {//BEGIN LESS THAN HALF START SIGNAL
        mw_iStartFlag = 1; // initialize startFlag to 0

        if(m_bBundle)
        {
          m_sMsgSensorStamp=m_pClock->GetTimeStamp();

m_sMsgSensor.Format(",%s,%s,WTKSENSOROK\r\n",m_sMsgSensorStamp,m_sMsgSensorStamp);
          m_bSensorHeard=1;
          m_bOverrideConcatFlagProfile = TRUE;
          m_bOverrideConcatFlagRut     = TRUE;
          m_bOverrideConcatFlagGyro    = TRUE;
          m_bOverrideConcatFlagDebug   = TRUE;
        }
        else
        {
          ReportStartSignal();
        }

      }//END LESS THAN HALF START SIGNAL
```

```
//==============================================================================v
//
// Profile
//
//==============================================================================v
```

PUT DISTANCE
PUT ACCELEROMETER
PUT LASER

```
    //=================================================================
    //
    //   Feed the data into the Walker Ride Machine
    //
    //=================================================================
    wsmPutDistance    ( m_pWsm, m_lColDistance   );
    wsmPutLasDataLeft ( m_pWsm, m_lColLaserLeft  );
    wsmPutLasDataRight( m_pWsm, m_lColLaserRight );
    wsmPutAccDataLeft ( m_pWsm, m_lColAccelLeft  );
    wsmPutAccDataRight( m_pWsm, m_lColAccelRight );
```

CHECK COUNT

```
    //=================================================================
    //
    //   Advance the counter and check the DMI Signal
    //
    //=================================================================
    m_iWsmCount = wsmCheckState(m_pWsm);
```

IF COUNT != -1

```
    //=================================================================
    //
    //   If the Count is not -1, then the DMI signal has completed a wave
    //
    //=================================================================
    if(m_iWsmCount!=-1)//BEGIN COUNT CHECK
    {

      if(m_bSGActive)
      {
```

STOP AND GO TEST: CHECK LIMITS AND SET FLAG

```
      //=================================================================
      //
      //   See if the count falls within the Stop and Go limits
      //
      //=================================================================
      if (m_bSGToggle==FALSE)
      {
        if (m_iWsmCount>m_iSGLimitStop)
        {
          m_bSGToggle=TRUE;
        }
      }
      else
      {
        if(m_iWsmCount<m_iSGLimitGo)
        {
          m_bSGToggle=FALSE;
        }
      }
```

IF FLAG=TRUE

```
      //=================================================================
      //
      // Stop and Go Phase 1: Adjust the Filter
      //
      //=================================================================
      if (m_bSGToggle)
      {
```

STOP AND GO PHASE 1: ADJUST THE COUNT

```
        iSGCount=SGPhase1(m_iWsmCount,m_iSGLimitGo);
        wsmSetCount(m_pWsm,iSGCount);
      }
```

CALCULATE PROFILE

```
      //=================================================================
      //
      // Calculate the Profile
      //
      //=================================================================
      ProfileLeftDLL  = wsmCalculateProfileLeft ( m_pWsm );
      ProfileRightDLL = wsmCalculateProfileRight( m_pWsm );
```

IF FLAG=TRUE

```
      //=================================================================
      //
      // Stop and Go Phase 2: Call the Slew Routine
      //
      //=================================================================
      if (m_bSGToggle)
      {
```

STOP AND GO PHASE 2: SLEW

```
        ProfileLeftDLL =SGPhase2(ProfileLeftDLL ,&m_lSGLastProfileLeft
,&m_bSGFirstTimeLeft ,m_lSGDifferenceLeft ,m_lSGCorrectionLeft);

ProfileRightDLL=SGPhase2(ProfileRightDLL,&m_lSGLastProfileRight,&m_bSGFirstTimeRight,m_lS
GDifferenceRight,m_lSGCorrectionRight);
      }

    }
    else
    {
      //=================================================================
      //
      // Calculate the Profile
      //
      //=================================================================
      ProfileLeftDLL  = wsmCalculateProfileLeft ( m_pWsm );
      ProfileRightDLL = wsmCalculateProfileRight( m_pWsm );
    }
```

RECORD PROFILE

```
      //=================================================================
      //
      //  Send Profile
      //
      //=================================================================
      ProfileInterpolate(ProfileLeftDLL,ProfileRightDLL);


      //=================================================================
      //
      // Reset for next DMI Signal
      //
      //=================================================================
      wsmReset(m_pWsm);
    }//END COUNT CHECK
```

The flow diagram for data acquisition and profile computation is illustrated in Figure 3.6. Also illustrated within the flow diagram, is the computation of the time or frequency function used for computing the phase one Stop and Go function. As discussed earlier, the distance, accelerometer, laser, and start sensors are sampled at typically 2000 Hz by the A/D module. The values are then placed or put into the profile computation module. The time sampled points are summed and once a complete cycle of the distance sensor is sensed, it is then averaged over each distance interval. The number of samples per distance sample is the variable FT (mw_f2). For Phase 1, the vehicle speed is computed and if the speed is below the SAG Start limit, e.g., 18 MPH, the variable FT (mw_f2) is replaced by the m_iSGF2LimitStop or maximum number of counts allowed for the stopping condition. This lower limit is used until the count exceeds the start limit at which time the actual sample count is once again used. In the next section, the methods of enabling and controlling the various parameters of phases one and two are discussed.

Controlling the Stop and Go Algorithm in WinTK Server

The enabling and controlling of the Stop and Go Algorithm in WinTK Server is controlled by use of the configuration commands applied to the server before data collection. The commands are listed below and include: A switch to turn on and off the Stop and Go algorithm for the Profile data, the speed at which the SAG algorithm (Window) begins, the speed at which it ends, the rate of change before applying the slew and the maximum slew change applied

Stop and Go Dictionary Entries

Configuration Tag used in CONFIG command: SGACTIVE
Use: Profile Stop and Go
Variable: CStopAndGoConfig.m_bActive
Units: N/A
Type: Boolean
Values: {True, False}
Default: True
Example: "CONFIG SGACTIVE TRUE\r\n"
Definition: A switch to turn on the Stop and Go algorithm for the Profile data.

Configuration Tag used in CONFIG command: SGSPEEDSTOP
Use: Profile Stop and Go
Variable: CStopAndGoConfig.m_dSpeedStop
Units: miles per hour
Type: double
Values: values greater than 0
Default: 12.0
Example: "CONFIG SGSPEEDSTOP 12.0\r\n"
Definition: The speed at which the Stop and Go routine begins, and affects the Profile data.

Configuration Tag used in CONFIG command: SGSPEEDGO
Use: Profile Stop and Go
Variable: CStopAndGoConfig.m_dSpeedGo
Units: miles per hour
Type: double
Values: values greater than 0
Default: 18.0
Example: "CONFIG SGSPEEDGO 18.0\r\n"
Definition: If the Stop and Go routine is running, this is the speed at which the Stop and Go routine ends.

Figure 3.6 Configuration Dictionary Entries for the SAG routine

Configuration Tag used in CONFIG command: SGDIFFERENCE
Use: Profile Stop and Go
Variable: CStopAndGoConfig.m_dDifference
Units: mils
Type: long
Values: values greater than 0
Default: 100
Example: "CONFIG SGDIFFERENCE 100\r\n"
Definition: During the second phase of the Stop and Go routine, a measure of the change of the profiles is detected. This measure of change is the difference.

Configuration Editor Key Word:
Configuration Tag used in CONFIG command: SGCORRECTION
Use: Profile Stop and Go
Variable: CStopAndGoConfig.m_dCorrection
Units: mils
Type: long
Values: values greater than 0
Default: 10
Example: "CONFIG SGCORRECTION 10\r\n"
Definition: During the second phase of the Stop and Go routine, a correction value is needed to offset the jump in the profile value.

Figure 3.7 Configuration Dictionary Entries for the SAG routine (continued)

In the next chapter, results of using the SAG function on the Austin Test Sections and on road sections around a section of city streets in Austin are discussed.

CHAPTER 4
Stop and Go Section Runs

A version of WinTK Server was made which implemented the two phases of the Stop and Go algorithm and which could then be used on the TxDOT Profiler. Runs were made on the Austin Test Sections as well as around the TxDOT Design Division, Building 37 on Bull Creek Road in Austin. In this chapter, comparisons are made between IRI and NSI values with and without Stop and Go enabled.

SAG operations on Austin Test Sections

Repeat runs were made with and without SAG operations on Bull Creek and the first two 0.1 mile sections of each of the Austin test sections. Two runs without SAG were first made on Bull Creek, Pearce Lane, Blue Bluff and FM3177 which can be used for comparison with the SAG runs. Three repeat runs were then made on each of the sections with SAG. For each of the SAG runs the vehicle was run at 35 MPH on Bull Creek and 50 MPH on the other sections. For each 0.1 mile section the vehicle was brought to a complete stop at the approximate middle of each 0.1 mile section. The vehicle would then accelerate in a manner typically done after stopping in an urban area, to the speed before beginning the stopping operation.

Table 4.1 lists the SI results from each of the runs. The modified WinTK server and the TxDOT Vamos Client Programs were used for data collection. Figure 4.1 illustrates the start of the Pearce Lane SAG run as taken from the camera on the TxDOT profiler. For all runs, the Stop and Go configuration setting were set to 12,20,80,6. That is, WinTK was sent the following configuration settings:

1. SGSPEEDSTOP = 12, The speed at which the Stop and Go routine begins, and affects the Profile data.

2. SGSPEEDGO = 20

If the Stop and Go routine is running, this is the speed at which the Stop and Go routine ends.

3. CONFIG SGDIFFERENCE = 80

During phase two of the Stop and Go routine, a measure of the change in the profiles is detected. If this change is greater than 80 (for this case), the difference will be corrected to the parameter described next, 6.

4. CONFIG SGCORRECTION = 6

The correction value used to offset the jump in the profile value.

The differences between the section averages with SAG vs without SAG are given in Table 4.1. As can be noted, the average SI differences between straight through, or runs without stopping in the section are listed. These SI average differences between measurements without SAG operations and those with SAG operations are around 0.1 SI.

Table 4.1 Stop and Go Runs on Bull Creek and Austin Test Sections

SI SAG Settings 12,20,80,6						
Run Type	**Run No.**	**0.1 mile**	**Pearce Lane**	**Blue Bluff**	**FM3177**	**Bull Creek**
St Thru	1	1	4.4	2.7	3.6	3.1
		2	4.5	2.9	3.5	3.5
St Thru	2	1	4.5	2.8	3.7	3.1
		2	4.5	2.9	3.5	3.5
SAG	1	1	4.2	2.7	3.6	3.1
		2	4.3	2.8	3.5	3.6
SAG	2	1	4.2	2.7	3.5	3.0
		2	4.4	2.9	3.4	3.6
SAG	3	1	4.3	2.8	3.5	3.0
		2	4.3	2.8	3.4	3.7
Avg Diff of St Thru vs SAG		1	0.2	0.0	0.1	0.1
		2	0.2	0.1	0.1	-0.1

Figure 4.1 Beginning of SAG Run on Pearce Lane Test Section

In order to simulate the conditions which could be expected when making profiler runs in an urban area with traffic lights, and numerous SAG operations, profiler runs with SAG enabled were made in an approximate three and a half mile loop around the Design Division offices on Bull Creek. Although each run had different SAG conditions, depending on traffic lights, construction, and other conditions, the same streets and lanes were taken. Figures 4.2 to 4.7 provide the left and right wheel paths of the runs. Although it was not possible to get runs without stops for the control sections, it was observed that none of the profiles shown had the wide profile swings as illustrated in Figure 1.1 and SI readings were in normally expected ranges.

Figure 4.8 illustrates a report obtained from TxDOT Proview Program for the SI readings from the three SAG runs.

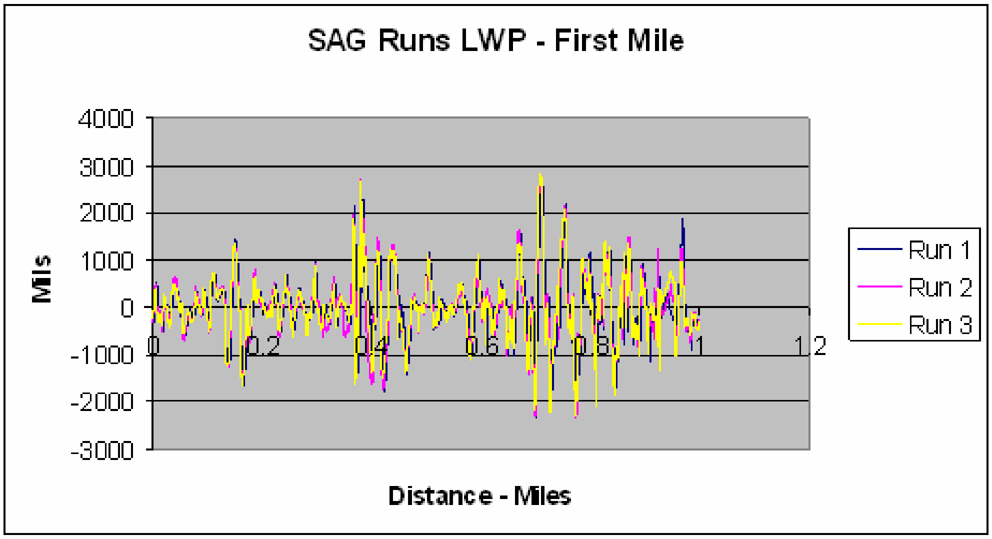

Figure 4.2 SAG Profile for LWP First Mile

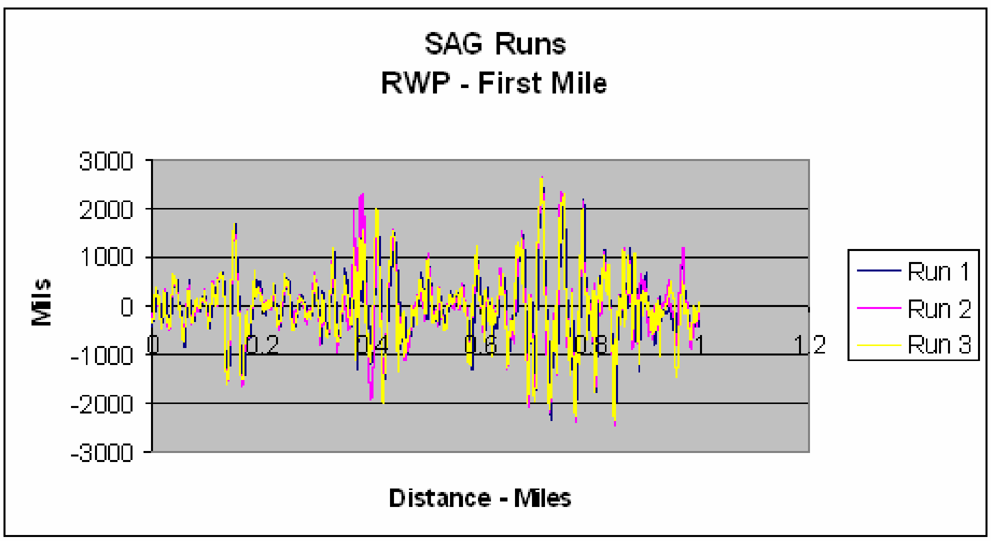

Figure 4.3 SAG Profile for RWP First Mile

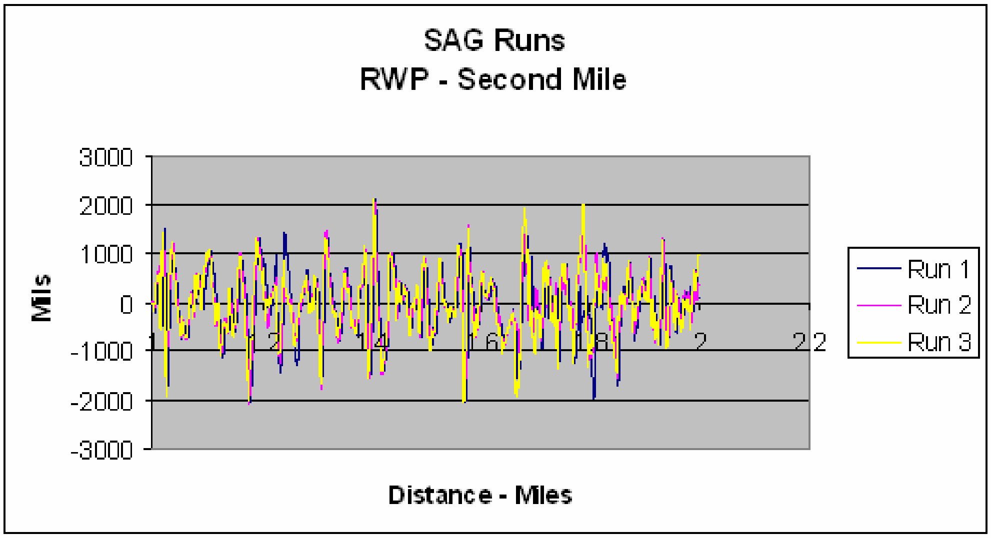

Figure 4.4 SAG Profile for LWP Second Mile

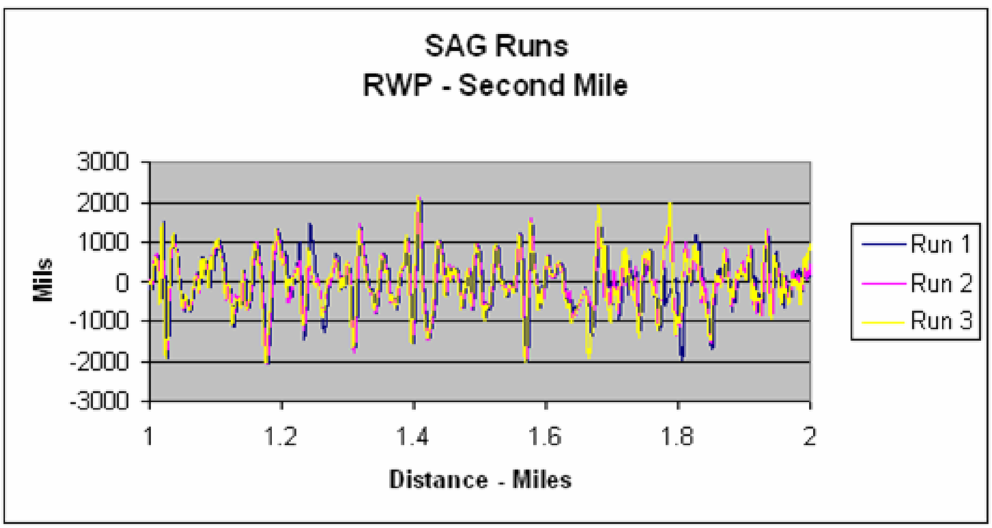

Figure 4.5 SAG Profile for RWP Second Mile

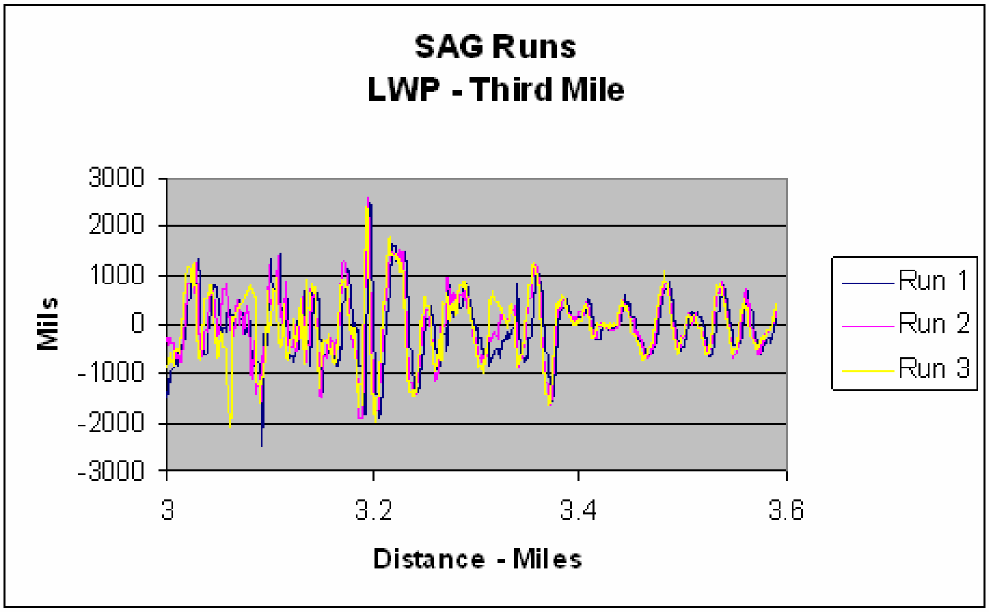

Figure 4.6 SAG Profile for LWP Third Mile

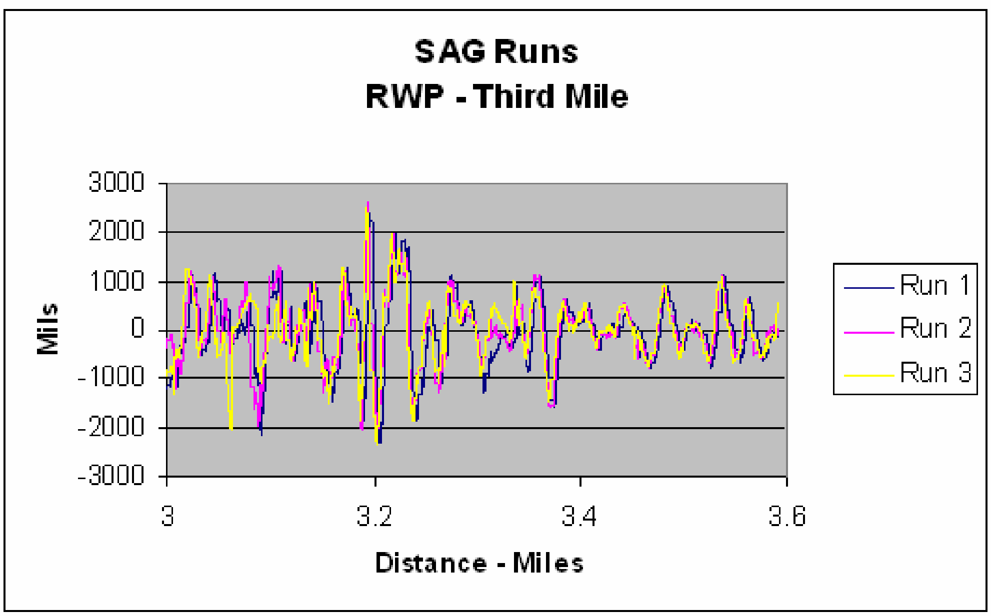

Figure 4.7 SAG Profile for RWP Third Mile

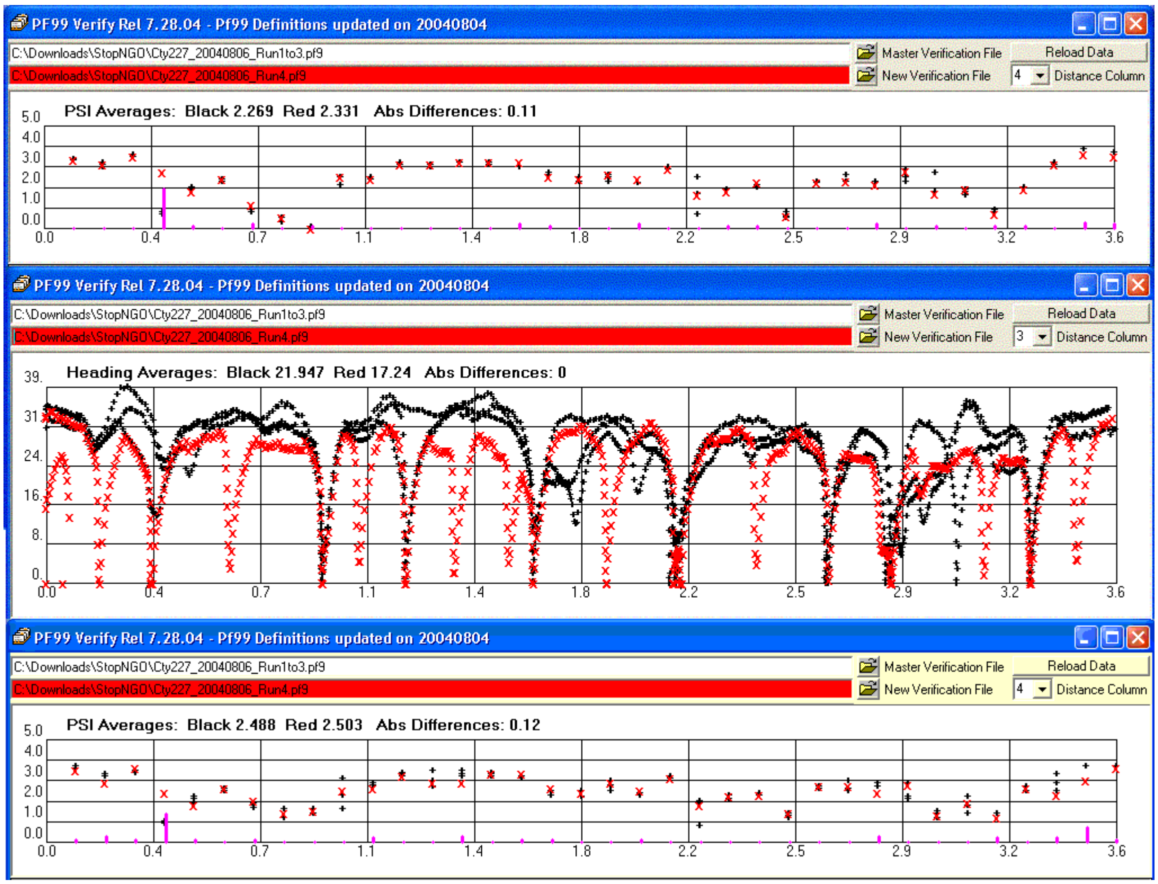

Figure 4.8 VAMOS Average SI Report for SAG 3.6 Mile Run

CHAPTER 5
Summary and Conclusions

As noted in Chapter 1, inertial reference profilers, such as TxDOT's profilers, are adversely affected by Stop and Go operations. Such operations commonly occur during PMIS data collections. When the profiler slows below 12 mph or comes to a complete stop, profile measurements are not valid. This situation causes the loss of ride data at intersections, in high traffic conditions, and in urban areas. In this research a method was developed that minimizes the affects of Stop and Go for Texas Profilers. In tests performed on the Austin Test sections the differences between SAG enabled and not enabled SI readings were on the average within 0.1 SI. Additional continuous runs were made around TxDOT Design Division Offices on Bull Creek in Austin with numerous SAG occurrences and no wide profile swings were noted when the SAG procedure was enabled.

The wide swings in profile due to SAG operations results in not only invalid measurements for the specific section which with the SAG operations, but also the adjacent section as the filter has to settle. The procedure developed prevents these swings thus resulting in correct measurements in the following section.

During the last year of the project TxDOT switched from the commercial distance encoder used and began manufacturing an in-house version. This resulted in a delay in implementing the final version of the SAG procedure. The profile and SAG algorithms can be significantly affected by errors in the distance encoder. In the case of the TxDOT in-house sprocket encoder, the circular distance between sprockets was not identical. Since the number of sprocket counts per mile is used to compute the distance between sprockets, the inaccuracy of the individual values affected the algorithms. Later versions of the sprockets resulted in more symmetrical units. Profile computations depend on accurate distance encoder readings, laser displacement and accelerometer readings.

The SAG procedure can be enabled or disabled by simply changing the configuration setting in VAMOS. Four other SAG settings are used for controlling the algorithm, and are described in Chapter 3. Proper SAG operations can begin by enabling the method and using the default configuration settings. Chapter 3 also contains the software statements used in WinTK for implementing the algorithm. UTA has given TxDOT the source code to the WinTK Server program and exclusive rights for its use. The SAG code is included in the server program.

It is recommended that TxDOT begin implementing the SAG procedures for PMIS data collections. During its implementation, the SAG configurations can be adjusted as needed.