TIME STUDY OF HARVESTING EQUIPMENT USING GPS-DERIVED POSITIONAL DATA

BY

Tim McDonald
Research Engineer
USDA Forest Service
Auburn, AL, USA

ABSTRACT

The objectives in this study were to develop and test a data analysis system for calculating machine productivity from GPS-derived positional information alone. A technique was used where positions were 'filtered' initially to locate specific events that were independent of what actually traveled the path, then these events were combined using user-specified rules into machine functions. The system was successful in producing gross-time-study data, but less so in providing detailed elemental times.

INTRODUCTION

Tie study is a basic tool used in studying the effects of management factors on productivity of logging systems. It has been used for many years in calculating costs for logging practices (Gardner 1963), and is fundamental in the analysis of forest operations. Collecting time study data, however, can be an expensive process (Olsen and Kellogg 1983). Time study of forest operations equipment ordinarily requires considerable travel to job sites. Forest operations tend to be dispersed across a large site, requiring several people to monitor activities scattered throughout the unit. There are many potential safety hazards working around woods equipmenf requiring well-trained, careful field crews. Although field coilection of data has been successful in the past, technology has evolved to the point that perhaps it 'is time to consider replacing the risk and cost of field crews with autonomous data collection systems.

Time study of forest harvesting systems common to the southern US is a relatively simple process. Especially for skidding, there are relatively few time elements to capture, and function is highly correlated with where the equipment is located on the site. Given the location of key areas, such as the deck or a delimbing gate, elemental times for skidding can be inferred almost entirely from where the skidder is relative to these positions. Automation of time study for these machines, therefore, may require only position data, which are simple and inexpensive to collect using Global Positioning Systems (GPS), and can be acquired remotely.

Although skidding is the simplest case in measuring production from position, it is by no means the only one. Any method for autonomous production studies would have to be flexible enough to analyze a number of different types of machines performing several different functions. Many of these functions would likely be unrelated to any fixed position. To fully capture sign&ant time elements of the production cycles, any autonomous time study system would need to be scalablein order to accept input data from a number of alternative sources.

The purpose of this research was to investigate methods for continuous autonomous monitoring of harvest system productivity. Initial efforts have focused on the use of positional data alone to infer time study information for forest harvesting equipment. Successful development of such a system would provide a cost-effective means of studying the efficiency of harvesting systems for long periods of time, providing detailed insight into the effects of management, equipment, and site factors on harvest efficiency. Specific objectives of this phase of the study were:

1) Develop methods and software for analyzing the movements of forest harvesting equipment to calculate productivity.

2) Use the developed system to perform time study on a skidder.

This paper describes the development and design objectives of a software analysis system for converting GPS-derived positional data into productivity information. The data requirements for the system are outlined, and an illustration of its use in calculating skidder productivity is presented.

EXPERIMENTAL METHODS

Tie study of forest harvesting equipment normally involves breaking the work cycle of a given machine into discrete tasks, or elements. Each of these elements has a time of performance associated with it, and perhaps some other parameter, e.g. distance traversed or volume of timber handled. In the field, elements are timed relative to specific starting and ending events. A skid cycle, for example, might be broken down into a *(1) travel empty, (2) position and grapple,* and (3) travel *loaded* sequence. Each of these elements has a characteristic event that marks its beginning and end: travel empty begins when the skidder leaves the deck and ends when the skidder stops prior to backing up to a load.

To perform time studies, an automated system would require some way of identifying the time and location of specific, individual events, and then be able to combine those events into sequences meaningful to the machine's function. Our system for deriving time study information from positional data was implemented using two components: 1) a 'feature' extraction sub-system to identify basic events in a machine path, and 2) an 'event processor' to combine characteristic movements and sub-events into machine-specific functions. The approach used a small set of features measured from a GPS-derived path that were independent of what actually created the motion to describe the function of a machine.

Language recognition is a useful conceptual model for understanding the time study system development. The path itself was filtered, much as speech might be filtered, to produce a sequence of fundamental events, or 'words' in the language recognition context. These events were then parsed using a specific grammar to form 'sentences', or sequences of events, describing the machine functionally. Users of the system decided what events, or 'words', were important to describe the function of a particular machine, then how those 'words' were arranged into meaningful combinations.

A skidder represents perhaps the simplest example of how this approach can be used to derive time study information. The fundamental events, or 'words', important in describing skidder function might include things like 'leaves deck', or 'enters deck', or 'reverses direction'. Combinations of these words, e.g. 'leaves deck; enters deck' would represent more complicated structures, in this case perhaps a 'skid cycle'.

The time study analysis system developed for this project was an implementation of this approach: a GPS-derived path was filtered to extract important basic features of the path, then a user-defined grammar was used to further filter, or 'reduce', the fundamental events into machine functions. The two steps were implemented as two computer programs: (1) `eventmap` to filter a path and extract interesting events; and (2) eventparse to combine events into machine functions.

To summarize the above, the process for reducing positional data to productivity information involved two stages of analysis. The first stage identified machine-independent features of the recorded motion, and the second collected those features into actions using a description of the machine's 'behavior'. These steps were implemented as two C++ programs, `eventmap` and eventparse, each requiring an input file with syntax outlined in the following. The analysis was completed when no more event parsing rules could be matched, after which the final list of events was output along with event times and total distance covered.

The 'fundamental events' extracted from the raw GPS data were from a set of measurable features that included the following:

Enter or leave a polygon
Occupy a location (Inside or outside a polygon)
Cross a linear feature
Start or stop moving
Reverse direction

Note that these features are independent of what actually created the path. This made it possible to derive the fundamental events for describing the actions of any machine using the same software tool. This

software tool (eventmap) read as input a machine path, and a specification of which events to filter out and report on. The syntax for describing fundamental events to be identified was like the following:

Listing 1.

```
polygon deck [
   (625432.978,3619740.551) ,
   (625394.501,3619759.311) ,
   (625366.377,3619699.622) ,
   (625391.118,3619671.207) ,
   (625442.670,3619673.970) ,
   (625449.781,3619720.626) ]

enter deck : enter-deck ;
leave deck : leave-deck ;
```

This input specification told eventmap to look for occurrences when the path entered or left the polygon deck, defmed by the UTM coordinates in the first 7 lines, and to generate a report of those occurrences by ejecting enter_deck or leave_deck identifier strings. The following is an example output:

Listing 2.

```
leave deck      18      30.06
enter-deck      16      15.00
leave-deck      149     225.78
enter-deck      70      145.32
leave-deck      28      43.39
enter-deck      69      142.52
leave-deck      79      115.62
enter_deck      62      149.22
```

The numbers following the leave-deck and enter_deck events were the accumulated time (in seconds) and distance (in m) since the previously occurring event.

Features were aiso specified as being logical combinations of other events, as in:

Listing 3.

```
reverse and not (inside deck): reverse-not-on-deck ;
```

The identified features were subsequently passed through the program eventparse, which reduced the machine-independent events into a set of work cycles. The parsing was done according to a user-specified grammar, or ruleset, that was defined using syntax similar to that used in regular expression matching. The following is an example:

Listing 4.

```
skid-cycle: leave-deck enter-deck ;
```

The above was the simplest form of event reduction. The parser looked for a leave-deck event followed by an enter-deck event and reduced that sequence (accumulating time and distance parameters) to a skid-cycle event.

More complicated rules were developed using reguiar expression operators. Regular expressions use a fixed syntax to match text input. A few characters are considered 'special' and provide a means of generating completion and alternative matching statements. A good introductory guide to regular expressions can be found at www.isys.net/susehilf/gnu/emacs/Regexps.html. Most regular expression

operators were implemented in the event parsing system, although the meaning of a few (notably the '^' and '$' operators) was changed to accommodate particulars of this application. An example of a parsing rule using the regular expression syntax follows:

Listing 5.

```
delimb: cross-gate-line \(\(reverse \)\|
    \ (cross-gate-line \) \) *cross-gate-line* ;
```

This rule instructs the parser to look for sequences of zero or more motion reversals or line crossings occurring between two cross_gate-line events and reduce that combination to a delimb event.

The analysis system was tested using data collected from a skidder (Timberjack 460C) working in an in-woods, clean chipping operation. The skidder was equipped with a single-arch grapple (Esco 2.8 m) and 34X3 1 .OO tires. The GPS receiver was a Trimble Pro XR model with data collected as a line feature in overdetetmined 3-D mode at 1 or 2 second intervals. Data were downloaded daily, differentially corrected, then exported to Arc/Info 'generate' format. Traffic data were collected over a period of 2 days of operation, totaling 29 observed skidder turns, where a turn is defined as one round trip from the deck, returning with a load of logs.

RESULTS AND DISCUSSION

Figure 1 shows the GPS traffic data for a single skidder turn, illustrating several points about the process of reducing this into machine production information. The simplest form of time study, and the easiest to implement using our analysis system, was to calculate gross production information. The only requirement in this case was to note the times when the skidder left and entered the deck area, and accumulate the distance in between those events. In terms of the analysis system, the process was to scan the traffic records for leaving and entering the deck events, then sum up the times and distances. The GPS-derived path was first filtered using the eventmap program and the feature specification of Listing 1,the result appearing iikethetextin Listing 2. The next stepwastouse the eventparse program toreducethese features to skid cycles, using the grammar defined in Listing 4.The final result of this simplest-case analysis would be a list of all reduced events, in this case looking like:

```
skid cycle   34    45.06
skid-cycle   219   371.10
skid_cycle   97    165.91
skid_ cycle  141   264.84
```

Numerical values were the time and distance (s and m) since the previously occurring event.

There was a flaw in this approach related to how the parameters were defined, i.e. cumulative time and distance since the last event The skid-cycle definition accumulated the time from both the enter-deck and leave_deck events. The leave_ deck event parameters were, by definition, the amount of time spent and distance traveled on the deck itself (time and distance since the last enter-deck. To exclude time spent on the deck from a skid-cycle, we could simply not accumulate the distance from the leave-deck event, which in terms of our analysis system could be done by modifying our previous event parsing specification (Listing 4) to the following:

```
on_deck-: leave-deck ;
skid_cycle: on-deck enter-deck ;
```

The difference was that the leave-deck event was simply renamed to on-deck, with the additional operator - causing the parameter values for that event to be set to 0. In this way, only the time spent off the deck was accumulated into a skid-cycle.

This was the simplest case for analyzing production data, and it was relatively straightforward to accomplish. The analysis became much more complicated, however, when trying to derive elemental times from the positional information alone.

To find elemental times, we first had to define some characteristic pattern from the motion data that signaled the event had occurred Refer, for example, to Figure 1, where the inset box contains a larger scale view of the terminal end of the skid cycle. Based on observation of the working patterns of this and other logging crews, the standard means of grappling a load of logs using a skidder was to drive to the bundle's vicinity, stop the skidder and back up to the butt end of the logs. The load was then secured using the grapple and hauled back to the deck. This pattern is easily observed in the traffic data from Figure.1 . The skidder stopped near the log bundle, reversed direction, then reversed direction again when starting back towards the deck Based on this assessment, a couple of basic elemental times were defined in the following way:

```
travel empty: leave-deck reverse$ ;
position-and-grapple: travel-empty" reverse reverse ;
travel-loaded: position-and-grapple" enter-deck ;
skid-cycle:  travel-empty position-and-grapple travel-loaded ;
```

In this example, the reverse in the travel-empty defmition was followed by the operator $. This operand told the eventparse program to 'push' the marked event back into the event stream. This syntax allowed the user to define events of interest in terms of when they occurred relative to other events. In the reduction of these events, a leave-deck reverse sequence was replaced by the new sequence travel-empty reverse. The same applied for the operator ^, only the retained event (travel_empty, for example, in the position_and_grapple definition) was prepended to the reduced event's name.

Using the definitions outlined above, simple skid cycles, along with some basic elemental times, could be extracted from the positional data. Skidder traffic, however, was rarely as simple as defined above. Referring back to Figure 1, for example, the traffic pattern was predictable until after the load had been grappled. Soon after starting towards the deck, however, the operator apparently stopped to pick up another load of logs, then proceeded towards a rub tree for delimbing. This logging operation used a flail delimber/debarker to produce clean chips, a system *that* leaves a large amount of residue *on the* deck. The skidders were ineffective in hauling these small residues back into the stand and they tended to accumulate heavily, causing congestion on the deck. To partly remedy this problem the skidder operators often used a rub tree out in the stand to partially delimb their loads before hauling them to me deck A characteristic pattern of several motion reversals when delimbing was apparent in the close-up view shown in Figure 1. In order to define a 'delimbing element, we had to distinguish direction reversals near the rub tree from those associated with grappling the load.

Distinguishing what motion reversals were associated with which activity required some other kind of information. The four small lines shown crossing the skidder path between the deck and the outer end of the skid cycle in Figure 1 served as 'landmarks' to be used in defining events. Deliibing, in this instance, was defined as motion reversals that were associated with crossing the 'line 4' landmark Similarly, other events were defined using the line crossing events as markers. Travel empty was distinguished from travel loaded by the order in which lines were crossed. Skid turns where multiple loads were pulled and dropped without entering the deck could be distinguished as well.

Listing 6 shows the final event parsing grammar used in reducing the measured GPS data to production cycles. The complexity of the specification reflects the unpredictable nature of the skidder movements. In order to distinguish elemental times, many different sorts of skid cycles were defined Some of this was due to poor placement of the lines used as landmarks for events. The lines used were physically laid out on the site using a GPS receiver after harvesting was completed. Using lines drawn directly from the data itself, or more landmarks, might have made the data reduction process simpler.

Performance of the analysis system was good in general. All 29 recorded skid cycles were successfully identified. The reported time and distance values corresponded well with measurements made directly from the GPS data using a GIS: the average difference between GIS-measured and calculated skid distances was -0.09 percent, which was not significantly different from 0 ($P < 0.7$). Only two turns showed a difference

greater than 1 percent between GIS-measured and calculated turn lengths. Event parsing grammars used to reduce the first day's data were applied without modification in analyzing the second day, and we felt this was evidence that the approach was a flexible method of calculating time study data for this particular harvesting operation.

Overall, the developed system was very general in nature, meeting a design goal that it be useful in developing production data from any type of harvest system. The disadvantage of this generality, however, was that it was difficult to use. Creating the event parsing specification was a non-trivial process and required a great deal of trial and error. We felt there was more intervention on the part of the user required than anticipated, locating landmarks and knowing the intricacies of how the equipment operators worked. Although there was evidence that the system could be used for developing time study information on any harvest system, further experimentation is required to validate the techniques.

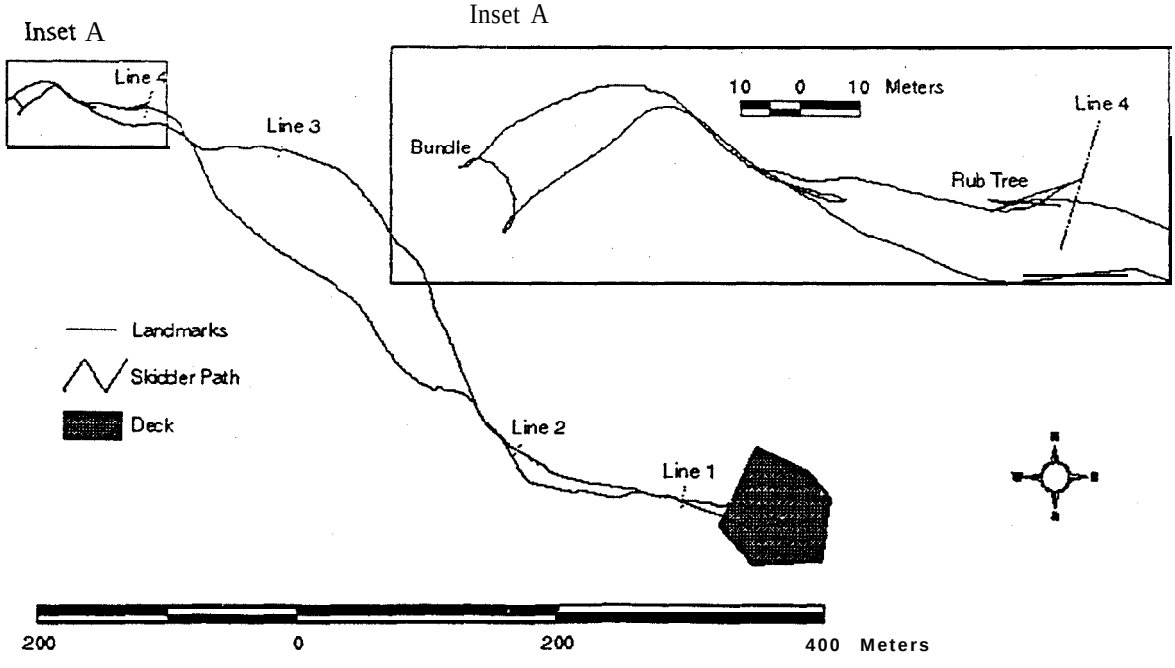
Extracting elemental times from the GPS data was felt to be only partially successful. Without an on-the-ground comparison of GPS-derived numbers with traditional time study data, we cannot conclusively say that cycle elements were reported accurately. Positioning and grappling events were especially difficult to distinguish using the GPS data analysis system. These events could be marked using other sensors - a pressure switch on the grapple, for example, to signal picking up or dropping a load. Because the data analysis system developed for this study was event driven, incorporating data from other sensors into the reduction process should be straightforward.

LITERATURE CITED

Gardner, Robert W. 1963. New tools to hone harvesting. Pulp and Paper. April 29: 73-75.

Oisen, Eldon D.; Kellogg, Loren L. 1983. Comparion of time-study techniques for evaluating logging production. Transactions of the ASAE. 1665- 1668; 1672.

Figure 1. Typical skid cycle as recorded using a GPS. The inset enlarges the positioning, grappling, and deliibing portion of the turn.



Listing 6. A listing of the final event specification grammar used to reduce path features to skid cycles.

```
#
# Anything after a hash mark is a comment
# Also, there are a couple of aliases used to simplify
# strict RE syntax, namely '{' and '}' can be used in
# place of '\(' and '\)', and ',' is the same as '\|'
#
# These rules are to eliminate some of the many
# instances of consecutive direction reversals
#
wiggle: reverse_nod^ {reverse_nod }+reverse_nod$ ;
exit_deck-: leave_deck
#
# Some cycle
#
delay%: stop_nod start_nod
travel_out: exit_deck^ {reverse_nod }?x_l1 {x_l2 }?
travel_out: exit_deck^ {reverse_nod }?x_l2 ;
woggle: x_l4^ {{wiggle },{reverse_nod },{start_nod },{stop_nod }}+x_l3$
;
woggle: x_l3^ {{wiggle },{reverse_nod },{start_nod },{stop_nod }}+x_l2$
;
woggle: x_l2^ {{wiggle },{reverse_nod },{start_nod },{stop_nod }}+x_l1$
;
woggle: x_l1^ {{wiggle },{reverse_nod },{start_nod },{stop_nod
}}+enter_deck$ ;
#
skid_cycle_1: exit_deck {travel_out }?{{wiggle },\
{start_nod },{stop_nod },{delay },{reverse_nod }}*enter_deck ;
#
travel_empty%: exit_deck travel_out reverse_nod {\
{wiggle },{reverse_nod },{stop_nod},{start_nod }} ;
travel_empty%: exit_deck travel_out delimb$ ;
#
position%: travel_empty^ {reverse_nod }*x_l4$ ;
#
delimb%: x_l4 {{x_l4},{reverse_nod },{wiggle },{stop_nod },{start_nod
}}*x_l3$ ;
delimb%: x_l4 {{x_l4},{reverse_nod },{wiggle },{stop_nod },{start_nod
}}*delimb ;
delimb%: x_l4 {{x_l4},{reverse_nod },{wiggle },{stop_nod },{start_nod
}}*x_l4 ;
delimb%: delimb {{x_l4},{delimb },{reverse_nod },\
{wiggle },{stop_nod },{start_nod }}*x_l4 ;
#
travel_loaded%: x_l4 {woggle }*x_l3 {woggle }*x_l2 {woggle }*x_l1
{woggle }*enter_deck ;
```

```
travel_loaded%: x_l3 {woggle }*x_l2 {woggle }*x_l1 {woggle }*enter_deck
;
#
# various versions of a skid cycle
#
skid_cycle_2: exit_deck {{wiggle },{start_nod },{stop_nod },\
{delay },{reverse_nod }}*travel_loaded ;

skid_cycle_3: travel_empty {{woggle },{reverse_nod },{start_nod },\
{stop_nod }}*{position }?{delimb }*{{reverse_nod },{start_nod },\
{woggle },{stop_nod }}*travel_loaded ;

skid_cycle_4: travel_empty {{wiggle },{position },{delimb },\
{start_nod },{stop_nod },{reverse_nod }}*travel_loaded ;

skid_cycle_5: travel_empty {{wiggle },{position },{start_nod },\
{stop_nod },{reverse_nod },{x_l4 }}*travel_loaded ;
#
# final pass
#
cycle_nolines_out: exit_deck {{position },{reverse_nod },{wiggle },
{stop_nod },{start_nod },{delimb }}*travel_loaded ;
#
intermediate_drop: x_l3 x_l2 x_l2 ;
inter_drop_loop: x_l3 x_l2 {{reverse_nod },{wiggle },{delimb }}*x_l3$ ;
inter_drop_loop: x_l3 x_l2 {{reverse_nod },{wiggle },
{delimb }}*intermediate_drop$ ;
#
multi_cycle: travel_empty {{position },{delimb }}*{{intermediate_drop
},
{inter_drop_loop }}*{{reverse_nod },{wiggle },{delimb }}*travel_loaded
;
```