The Freeway Service Patrol Evaluation Project: Database, Support Programs, and Accessibility

Karl Petty, Hisham Noeimi, Kumud Sanwal, Dan Rydzewski,
Alexander Skabardonis and Pravin Varaiya

California PATH Program
University of California, Berkeley
Berkeley, CA 94720

and

Haitham Al-Deek
Department of Civil & Environmental Engineering
University of Central Florida
Orlando, FL 32816

Abstract

This paper discusses the Freeway Service Patrol (FSP) Evaluation Project conducted under the California PATH program. The goal of this project was to determine the cost effectiveness of the FSP program along a ten-mile "beat" on I-880, near Hayward, California. We collected data from three different sources: loop detector data, probe vehicle data, and incident log data. We describe these sources in detail and the problems that were encountered with them. We also discuss the support programs developed to process this data, and the efforts to place the data and the programs online for use by researchers around the country.

1 Introduction

This paper describes various byproducts of the Freeway Service Patrol Evaluation Project conducted as part of the PATH program at the University of California, Berkeley. The Freeway Service Patrol (FSP) is a collection of tow trucks that drive around a particular section of the freeway during peak periods and provide free assistance to motorists with breakdowns or accidents. In our study section there were usually two tow trucks that continuously drove around a 10 to 12 mile section of freeway. The hypothesis is that since these tow trucks are already driving on the freeway, they will respond to incidents quicker than a tow truck called by a motorist from a roadside call box or a cellular phone. Therefore, they will reduce the duration of the incident and hence reduce congestion on the freeway caused by this incident.

Having tow trucks drive around the freeway and providing free assistance to motorists incurs some cost which is paid for, in this case, by various state, local, and federal government agencies. The benefit of reduced congestion is a reduction in delay which is measured in vehicle - hours. This benefit can be translated into a monetary value by multiplying the delay by the value of time per vehicle (in the state of California, the value of time per vehicle was taken as $10/hour). The resulting monetary value is the savings resulting from reducing the incident duration by some

amount. By comparing this benefit to the cost of the FSP tow trucks, a direct benefit to cost ratio can be computed. There have been several theoretical studies to determine the benefit to cost ratio for FSP type projects (Fambro, Dudek and Messer (1976), Wohlschlager and Balke (1992), and Morris and Lee (1994)). While the primary goal of the FSP Evaluation Project was to measure the savings in delay caused by the FSP, and hence the benefit to cost ratio, the main goal of this paper is to discuss the data collected during the evaluation project, the software developed to process that data, and the online facilities available to researchers on the Internet. A detailed analysis of the results of the FSP Evaluation Project, including the benefit to cost ratio, can be found in Skabardonis, Noeimi, Petty, Rydzewski, Varaiya and Al-Deek (1995).

The data collection effort produced perhaps the largest and most comprehensive database on incidents and freeway operating conditions to-date. The data processing and analysis software provides the integration of the various data elements and the utilization of the data for several functions in transportation management centers (TMCs) including, but not limited to, incident detection, traffic flow dynamics, and travel time estimation.

Section 2 of the paper describes the different types of field data that were collected during the study. Sections 3, 4, and 5 describe the data processing and discuss the various problems with the different data sets and the steps taken to resolve them. Section 6 presents the software written for data processing and analysis. Section 7 discusses what is available online and how researchers can go about getting the data and the software. Section 8 summarizes the study findings.

2 The Data Collected

In order to estimate the delay savings attributable to the FSP, data was collected over two time periods: once when the FSP was not in operation (the "before" period) and once when the FSP was in operation (the "after" period). The before study took place from February 16 through March 19, 1993 with the after study taking place from September 27 through October 29, 1993. All of the data was collected on a section of the I-880 freeway in Hayward, California. The study section was 9.2 miles long and varied from 3 to 5 lanes. An HOV lane covered approximately 3.5 miles of the study section. There were several sections that lacked right-hand shoulders and/or left-hand shoulders. Call boxes were installed at approximately 1/4 mile intervals. Data was collected on the weekdays during the peak periods (6:30 - 9:30 am and 3:30 - 6:30 pm). For each study period we collected three different types of data: loop detector data, probe vehicle data and an incident characteristics database.

2.1 The Loop Data

Loop detectors were placed approximately $\frac{1}{3}$ of a mile apart on the freeway mainline and on all the on- and off-ramps. There were a total of 322 mainline detectors, 18 on-ramp detectors, and 14 off-ramp detectors. These loops were connected to standard Type 170 controllers that recorded speeds, flows, occupancies and loop on and off times. The loop data was stored each weekday from 5:00am until 10:00am and then from 2:00pm until 7:00pm on the disk drive of the laptop computers that were installed in each 170 controller cabinet. Every few days these files were transferred to floppy disks and then to a Sun Workstation where the processing took place. Due to defective disks and intermittent power failures at the 170 controllers we were able to collect only 434 out of a possible 456 files for the before study (19 detector stations over 24 days), and 454 out of 475 files for the after study. Each file is approximately 1.7 Mbytes making a complete day approximately 33 Mbytes and the entire before study approximately 760 Mbytes.

A loop detector is just the standard inductive loop that is buried under the freeway that picks up the presence of a vehicle traveling over it as in Figure 1. The figure on the left side of Figure 1 is a graphical representation of one lane of a freeway.

On the main line lanes the detectors are placed in pairs, but on the on-ramps and off-ramps they are single detectors. Knowing the distance between the detectors, Δ, and the time taken by a vehicle to travel that distance, τ, one can calculate the average speed of the vehicles and the average occupancy of the loop detector per time period. The number of cars that pass over the detectors per time period, or the flow, is simply determined by counting the number of pulses.

2.2 The Probe Vehicle Data

Throughout the experiment four vehicles were driven around the study section at 7 minute average headways during the data collection period. For each probe vehicle, there were approximately six runs for each three hour shift. Although this should give us 1152 runs for the before study and 1056 runs for the after study, there were many breakdowns of the probe vehicles and the computers within them. Therefore, we were only able to collect 881 runs in the before study and 764 runs in the after study.

These vehicles had computers that recorded the car's movement and the driver's key presses and then saved these to various files on a PC floppy disk. There are a total of four data files from each car for each shift:

key.dat This file saves the keys that the drivers type in. The drivers type a key each time they start a run, pass a gore point (specific on-ramps or off-ramps), or pass an incident. Since the odometer readings are recorded at every key press, one expects that it will be possible to determine the location of the vehicle when the key was pressed and hence the location of the incident. Conceptually, these key presses correspond to the marked points in Figure 2, which has a picture of the section of the freeway that was studied.

fsp.dat Certain loop detectors in the study section were set up to emit a constant radio signal. The system that produced these signals is called the inductive radio (INRAD) system. The probe vehicles were equipped to detect and record these signals. The `fsp.dat` files contain an odometer reading and a time stamp for each time the car drove over one of the INRAD beacons. We had a total of three working INRAD beacons. The support software uses these location markers to pinpoint where the probe vehicle was at a particular time.

nav.dat Probe vehicles were all equipped with digital compasses. The PCs in the cars stored the odometer reading and the digital compass reading every second in files named `nav.dat`. With this data it is possible to plot the trajectories of all of the vehicles.

gps.dat The `gps.dat` file is the data from the GPS equipment in the car. It contains the latitude and longitude of the car which can also be used to plot the vehicle trajectory.

2.3 The Incident Data

The incident data is one database file with approximately 80 columns of information per incident. The database was collected during the experiment by the drivers of the probe vehicles. When they were driving on the study section and they passed an incident they would transmit via radio to a supervisor information about the incident characteristics (vehicle type, location, lanes blocked, presence of tow-truck or police vehicle, etc). All of this would be written down by the supervisor on a standard form. This form was then coded into an Excel spreadsheet with numerical entries in each

column so that a computer could process it later. It was then transferred to a workstation where it is stored as a tab delimited set of lines.

It is interesting to note that the breakdown of the incident characteristics agrees closely with the previously reported data of Lindley (1986). A breakdown of our incident characteristics averaged over the "before" and "after" study is given in Figure 3. The first number one each line is from the before study, the second number is from the after study and the number in parenthesis are from Lindley (1986).

3 Loop Data Processing

The loop data processing started off with the raw files from the 170 controllers stored in a compressed format on floppy disks. In the first step, these files were uncompressed and transferred to a Sun workstation. Once on the workstation, the support/analysis software converted the raw 170 output to ASCII readable text. During this conversion process the data is aggregated from a one second reporting time to a reporting time chosen by the user (usually 30 or 60 seconds). The software then attempts to correct a few of the problems associated with the loop data. At this stage, for a 30 second reporting time, the loop data takes up approximately 500 Mbytes of disk space.

Quite a few things turned out to be wrong with the loop data. In most cases we fixed these problems by programming the support software to take care of them. In other cases there is nothing that we could do and so we just noted them in the software manual (Petty (1995)) and left them. This section will explain these problems. This paper does not give a canonical list of which specific pieces of data are wrong. That information can be found in Skabardonis, *et al.* (1995).

Even though the loops are the best data source, several loop detectors were continuously broken, didn't record data periodically, or counted things incorrectly. There are two "fixes" that we attempt to do on the loop data: a "hole" fix, to fill in any missing data, and a "consistency" fix, to correct systematic errors in the loop data.

3.1 Loop Data Drop Outs

There are two different types of missing loop data. The first type is when a loop detector is gone for the whole day because either the computer was broken or the disk that the data was stored on for that day was bad, etc. The second type is when the loop detector doesn't report data for a period of time. What we do is to recreate the data for the missing detectors from the adjacent (adjacent in distance, not time) loop detectors.

There are a couple of ways to recreate the missing data. We could just copy the data from the adjacent upstream loop detector. Alternatively, we could average the values from the two adjacent loop detectors. The criterion that we use to govern how we recreate the data is that the calculated delay should be the same as if the loop detector was not there in the first place. To understand this approach, consider two situations indicated in Figure 4:

1. In the first situation, depicted on the left side of Figure 4, a loop detector is missing. The adjacent loop detectors expand the length of their segments to cover the missing detector. Based on these new lengths we calculate the delay for these two segments. This is the situation where the loop detector is not there in the first place.

2. In the second situation, on the right side of Figure 4, a loop detector is missing but we recreate some data for that detector. We now have three segments all with their original lengths. Based on the new data and the original lengths we calculate the delay for all three segments. This is obviously the situation where the loop data is recreated.

Since these two situations cover the same section of the freeway the total delay should be the same. This is the criterion that we use to figure out the formulas for recreating any missing loop data. These two situations are displayed graphically in Figure 4. The numbers on the top are the loop detector/segment numbers. Note that the segment lengths change from situation 1 to situation 2. The equation used for calculating the delay at a particular loop segment is:

$$D_k = L_k \frac{\Delta T}{60} F_k \left(\frac{1}{V_k} - \frac{1}{V_T} \right), \tag{1}$$

where D_k is the delay on segment k in vehicle-hours, L_k is the length of segment k in miles, ΔT is the time slice in minutes, F_k is the flow on segment k in vehicles per hour, V_k is the speed on segment k in miles per hour, and V_T is the threshold or congestion speed in miles per hour. We set the delays for the two situations in Figure 4,

$$D_{Situation1} = KF_1 \left(L_1 + \frac{L_2}{2} \right) \left(\frac{1}{V_1} - \frac{1}{V_T} \right) + KF_3 \left(L_3 + \frac{L_2}{2} \right) \left(\frac{1}{V_3} - \frac{1}{V_T} \right), \tag{2}$$

$$D_{Situation2} = KF_1 L_1 \left(\frac{1}{V_1} - \frac{1}{V_T} \right) + KF_2 L_2 \left(\frac{1}{V_2} - \frac{1}{V_T} \right) + KF_3 L_3 \left(\frac{1}{V_3} - \frac{1}{V_T} \right), \tag{3}$$

to be equal, where K is the time conversion constant. Next, cancel terms:

$$\frac{F_1 L_2}{2} \left(\frac{1}{V_1} - \frac{1}{V_T} \right) + \frac{F_3 L_2}{2} \left(\frac{1}{V_3} - \frac{1}{V_T} \right) = F_2 L_2 \left(\frac{1}{V_2} - \frac{1}{V_T} \right). \tag{4}$$

If we assume that this must hold for all values of V_T then we can pull out the terms that have V_T in them to get:

$$\frac{F_1 L_2}{2V_T} + \frac{F_3 L_2}{2V_T} = \frac{F_2 L_2}{V_T}. \tag{5}$$

Finally, we solve equation 5 for F_2 to get:

$$\frac{F_1 + F_3}{2} = F_2. \tag{6}$$

If we turn around and plug equation 6 into equation 4 then we get:

$$\frac{F_1 L_2}{2} \left(\frac{1}{V_1} - \frac{1}{V_T} \right) + \frac{F_3 L_2}{2} \left(\frac{1}{V_3} - \frac{1}{V_T} \right) = \frac{F_1 + F_3}{2} L_2 \left(\frac{1}{V_2} - \frac{1}{V_T} \right), \tag{7}$$

$$\frac{F_1}{V_1} + \frac{F_3}{V_3} = \frac{F_1 + F_3}{V_2}, \tag{8}$$

$$V_2 = \frac{F_1 + F_3}{\frac{F_1}{V_1} + \frac{F_3}{V_3}}, \tag{9}$$

So the equations used to recreate the data for loop detector #2 in situation 2 in Figure 4 are equations 6 and 9 above. Note that the distance drops out and that this will work no matter how many detectors in a row are missing. If the detector that is missing is at the end of our study section, that is, we don't have any data for one side, then the counts, speeds, and occupancies are simply copied from the side for which we do have data.

3.2 Inconsistent Loop Data

The loop data may have inaccuracies/inconsistencies due to a variety of reasons other than a complete loop malfunction (which are taken care of by the support software by filling in the holes). Some of these may lead to errors of the following nature:

Counts: The loops may not be sensitive enough to detect all the vehicles passing over them and thus may be undercounting. Also, vehicles not in the center of the lane may be missed as is the case of vehicles attempting to change lanes. On the other hand, loops may be detecting vehicles that are on neighboring lanes and hence overcounting.

Speeds: The resolution with which the time headways can be measured is 1/60 second. This limits the accuracy of measured vehicle speeds. Further, mismatch in tuning of a pair of coupled detectors leads to errors that cannot be compensated.

Occupancies: The detector thresholds may not be tuned properly and this can result in bias in the measured occupancies.

A subroutine was developed to identify these problems with the loop data and to generate a set of correction factors that can be applied to the loop data. When it is run, the support software program can read in these correction factors and fix the loop data so that it is consistent as described in the following subsections.

3.2.1 Occupancy Correction

The reliability of the densities obtained from the occupancy data provided by the detectors depends on proper tuning of the detector threshold. In the k^{th} interval, the speed $v[k]$, the flow per lane $q[k]$ and the density $\rho[k]$ are related by

$$q[k] = \rho[k] \times v[k].$$

The traffic density $\rho[k]$ is proportional to the percent occupancy, $occ[k]$, measured by the detector as $\rho[k] = K_d \times occ[k]$, where K_d is a constant that depends on the detector. Substituting this into the previous equation and taking logarithms, we obtain

$$\log(q[k]) = \log(K_d) + \log(v[k]) + \log(occ[k]).$$

From the field data we can find the least squares regression fit for K_d for each of the mainline detectors. These coefficients can be used to compensate the detector errors by using the estimated constant K_d to obtain the actual density.

3.2.2 Counts Correction

Since vehicles are neither created nor destroyed, we can apply conservation laws for vehicles. If we consider a section of highway which has detectors at the beginning and end as well as on the ramps, then the number of vehicles accumulated over the k^{th} data collection interval is

$$\rho[k+1]L - \rho[k]L = q_{in}[k] - q_{out}[k] + r[k] - s[k].$$

Where q_{in} is the mainline flow into the section, q_{out} is the mainline flow out of the section, r is the on-ramp flow, and s is the off-ramp flow. If the detectors on a section of the highway count correctly, then the accumulation should not have a drift (should remain bounded within reasonable numbers, depending on the length and number of lanes in the section) since this section of highway can only

hold a limited number of vehicles. We compute the average accumulation per minute over a long period (about 4 hours) and if its absolute value exceeds a set threshold, then our algorithm checks the consistency of the detectors associated with that section and compensation factors are computed as a fraction of the flow of the nearest mainline flow. Using these correction factors the software computes the flows that satisfy the consistency requirements.

4 Probe Vehicle Data Processing

Most of the problems with the car data arose during the data logging transactions by the drivers. As we will see later on, we used the probe vehicle data to attempt to correct the incident locations and durations. As explained in the subsections below, the only corrections that were made to the probe vehicle data was to identify bad data and weed it out.

4.1 Key Presses

Figure 2 in Section 2.2 shows the situations in which drivers were supposed to press keys when they were driving along the freeway section. A typical sequence of key presses that the program would expect is something like: loop start, southbound start, gore point, gore point, southbound end, northbound start, gore point, gore point, northbound end, etc. Often the drivers either forgot to press the keys or they pressed the wrong key. This causes problems:

- It is harder to match up the incident database with the car key presses. This makes it hard to figure out the precise location of the incident.
- When a driver doesn't radio in that they see an incident then it could cause an error in the incident duration. If they were the first or the last driver to pass an incident and they don't report it then the duration of the incident will be shorter than it should have been.
- In one routine, the support software attempts to calculate the travel time for the southbound run and for the northbound run based on the start and end keys. If the drivers don't press these keys properly (in the right order and at the right time) then we can't get accurate travel times.

It should be mentioned that during the study we determined that Car 2 was consistently bad. In the middle of operation it would make errors in the data stream that it was saving to the `nav.dat` file. Cars were changed in the before study once we figured out that something was wrong. This problem persisted through the after study as well. Since Car 2 was always bad, its use was limited to the radio reports of incidents.

5 Incident Data Processing

In order to accurately determine the delay attributable to a specific incident it is important to be able to determine the time an incident occurred and it's location accurately. Most of the processing that takes place with the incident data is done to determine these two quantities. The next two sections discuss why it is difficult to determine the incident location and starting time from the verbal reports of the probe vehicle drivers.

5.1 Bad Incident Placement

It turns out that sometimes the location of the incident is not accurate. We fix that by a routine that attempts to correlate the locations of the incidents in the incident database with the locations of the key presses in the car data. What we are trying is to match up two different sets of data: the incident database, which is just the observations of the drivers, and the data from the probe vehicles, which are just the computer generated `key.dat` files that we have for every car, for every shift. The expectation in doing this is that we can get a more accurate location for the incident.

There are problems with each set of data. In the incident database the location of the incident is only stored in very general terms like, "a half a mile before A-Street," or "3/4 of a mile past Winton." These distances are based solely on the driver's perception and in our experience they are quite inaccurate. When we look at the car data the situation doesn't get much better. Even though a key was pressed when the driver passed an incident, there is no link between a specific incident and a key press. There is only a marker in the `key.dat` file that says that a key was pressed when the odometer said a certain value. The odometer reading that is recorded is just the total distance from the start of the shift. So to find out the exact location on the freeway where the key was pressed we need to do a little more processing. This extra processing involves trying to determine where the vehicle was on the freeway based on other keys that the driver typed in.

The correlation routine attempts to merge the two data sets to get a more accurate incident location. To do this it first defines a box for each incident in the time-distance plane, as in Figure 5. This box is centered in distance around the location of the incident recorded in the incident database. The width of the box is the distance error bound that is a function of how accurate we think the data is. The box is also centered in time around the recorded time of the incident. The length of the box on the time axis can be increased by a user settable option.

Once the program defines all of the incident boxes on a space-time plot it places all of the key presses from all of the `key.dat` files for that shift and that day on the plot as well. An example of the key presses from the car data is given in Figure 6.

Figure 6 is a plot of all of the runs of a probe vehicle down the freeway and back. Since this is a plot of time versus distance, the inverse of the slope is the speed of the vehicle. The steep region in the middle of each run is where the car got off the freeway and turned around. Each "x" is a key press that corresponds to an incident. You can see that if you take a vertical line at an incident then it should pass through multiple key presses. These are just the different times that cars passed the same incident.

Finally, the correlation plot is made by superimposing all of the incident plots for one shift with all of the key press plots for one shift as in Figure 7. The key presses should fall inside of the incident boxes but quite often they don't. Once the computer has these plots stored in memory, the software attempts to figure out where the incident actually occurred. It does this by taking the average location of all the key presses that occurred inside of each box. This average location is called the incident location. If a key press doesn't fall within any box then it is not counted at all and there is no attempt to find the closest box that a key press could be in.

In some cases it was obvious that the placement of the incident was wrong. There were a few correlation plots that looked like the left plot in Figure 8. For these incidents, we figured out the amount of shift that was needed to place the incident box on top of the key presses. We then coded this shift into a file that can be read in to adjust the location of the incidents. Whether or not this file is read in to adjust the incidents is decided by the user at runtime.

Unfortunately the correlation between the incident data and the probe vehicle data gave marginal improvements. There are several reasons for this:

1. The driver could have pressed a key where there was no incident.

2. The driver could have not pressed a key when there was an incident.

3. The computer might have been broken in a car that reported an incident. This would cause an incident to appear in the database with no corresponding key presses.

4. The driver-reported location in the incident database could be wrong.

5. The support software might not be able to determine the location on the freeway of the key press in the `key.dat` file accurately.

However, we only need to know which loop detector was directly upstream of the incident. Although we spend a lot of time trying to adjust the location of the incident, in the end the only granularity that was needed was about 1/3 of a mile.

5.2 Bad Incident Duration

Another thing that should be corrected in the incident database is the duration of the incidents. Since we only witnessed incidents when a probe vehicle would drive by, we don't know when an incident started and ended. The actual incident duration is longer than the recorded incident duration as can be seen on the left side of Figure 9. In this picture the diagonal lines are the trajectories that the various probe vehicles made. The solid box corresponds to the duration of the incident that was recorded in the incident database and the dotted line corresponds to the actual duration of the incident. The incident actually occurred sometime between when we witnessed it the first time and when somebody drove by the same spot and didn't witness it. One way to estimate the true incident start or end time is to figure out the average time between any two vehicles and then add one half of this time to the starting time and one half to the ending time. The problem with this is that the headway time is not constant it has a distribution. Adding some constant to both sides of the duration might not be the best thing to do. What we could do instead is figure out the last time that a car drove by that didn't witness the incident and then take a certain fraction of this time. The fraction of this time that we usually use is 50% simply because there is no reason to think that start time of the incident isn't uniformly distributed between when we didn't see it and the first time that we did see it. This method should give a more accurate approximation to the duration of the incident. Once again, since processing the car data takes a long time we do the processing once and then saving the results to a file. This file can then be read in at runtime as a substitute for extracting this information from the car data.

Of course, doing this method brings up its own problems. The instrumentation in Car 2 was broken and we never got any location data from it. But since the driver was still driving around the freeway and reporting incidents, we have time entries in the incident database for a car that has no supporting probe vehicle data. If the time entries in the incident database had the car # that reported them then we could have overcome this problem. Since they don't, this discrepancy can cause errors in the incident duration calculation. For example, let's say that car 3 drove past an incident and witnessed it just as it was being cleaned up, and that the next two cars to pass the incident were car 2 and then car 1 and neither one of them saw the incident. The correct ending time for this incident should be half of the time between when car 3 passed the incident and witnessed it and when car 2 passed the incident and didn't witness it. But since we don't have any data from car 2 the **fsp** program thinks that the next car that passed by was car 1. Therefore the ending time for this incident will be inflated as can be seen on the right side of Figure 9.

This concludes the discussion of the various problems associated with the I-880 database: the holes in the loop data, and the location in time and space of the incidents. The next section will discuss the software tool developed to process this data and extract meaningful results.

6 Software Description

The software developed in this study consists of two main programs: **fsp** and **xfsp**. The **fsp** program is the software tool used to interrogate the data. This program performs diagnostics on the data, generates error reports, and makes plots of various pieces of data. The program takes as its input arguments a file that is called a runfile, an incident filter file, and an incident run number. The runfile contains all of the user settable options for the **fsp** program. The incident filter tells the program which incidents to filter out of the incident database and the incident run number is just an index for the output files.

The **fsp** program generates several types of output. If one were to run the program on a complete data set the program would take about 12 hours and generate up to 8000 files. A summary of the various types of output files and plots is given in Table 1. A complete description of the output and interpretation is given in Petty (1995).

Probably the most salient feature of the **fsp** program is that it was designed to be a loop data processing tool. It is not restricted to working with only our data set. It can take in any set of loop detector data and produce the same type of loop output (e.g., traffic density plots). There are plans to use this program in another FSP evaluation project at a different location in California.

The **xfsp** program is a graphical user interface to the **fsp** program. This program allows the user to generate the runfile and the incident filter by clicking on various buttons and widgets with the mouse. It will also collect the output for you and display it in a window on the screen. There is an extensive help system within **xfsp** that should allow the user to control the **fsp** program without much need for the manual (Petty (1995)). One can think of the **xfsp** program as being a "wrapper" for the **fsp** program: the **fsp** program is what does all of the processing work and the **xfsp** program is what deals with the user. One way to view this graphically is in Figure 10.

Figure 10 shows the **xfsp** program being in control of the **fsp** program: the **xfsp** program tells the **fsp** program what to do and then reads the output back from it. On the same figure is another program named **xfspview**. **Xfspview** is a program that allows the user to browse through the output of the **fsp** program by simply clicking on buttons. The **xfsp** program starts up the **xfspview** program every time that it runs the **fsp** program.

The **xfsp** and **xfspview** programs were written on top of a software tool called Tcl/Tk (Ousterhout (1993)). Tcl/Tk is a powerful command language that lets you manipulate graphical objects like windows, buttons, sliderbars, etc. The **xfsp** control window is given in Figure 11.

There are two distinct parts of the **xfsp** control window. The top part consists of a menu bar with five buttons in it that control the various administrative parts of the program like file manipulation, the main help screens, exiting the program, etc. The large window right below the menu bar is the data flow window. Each of the buttons in the data flow window has various options "underneath" it. If you click on one of the buttons then a window will pop up with the options specific to that button's function. For example, there is a button called "Fix Loop Data." If you click on that button then the window that holds the options that deal with fixing the loop data will pop up. Once you are done setting those options then you can pop that window down and play with some different options. For every button and every option there is either a help screen or an explanation button.

The layout of the data flow window was chosen to reflect the actual flow of data inside the **fsp** program. The arrows indicate how the data flows (which is mostly from left to right). The expectation is that this would help the user understand the function of each of the options.

7 What's Online?

One goal of this project was to provide the research community at large with all of the data that was collected and the software that processes this data. We have done this by placing all of the data on the Internet so they are available to any researcher with Internet access. There are three things that have been placed online:

1. The raw data (loop, probe vehicle, and incident database).
2. The processed loop data averaged over 1 and 5 minute intervals. The holes have been fixed and the consistency errors have been corrected.
3. The data processing and analysis software (*fsp*, *xfsp*, and *xfspview*).

One way that the data can be downloaded is by using anonymous ftp to connect to the server that holds the data: `www-path.eecs.berkeley.edu`. Table 2 gives a list of where on this server you can look to find the various software packages and data. Note that the different packages listed under **Support Software** in Table 2 are needed only if you are going to run the **xfsp** program. A complete description of these different packages is given in the manual for the **fsp** program (Petty (1995)).

Another way that one can download the data is through the World Wide Web using the program **netscape**, if you have access to a workstation running X, or by using the program **lynx** if you only have a character based terminal. **Netscape** is the standard World Wide Web browser program and it is used here as a user-friendly interface to download the data. The uniform resource locator (URL) of the FSP project is:

```
http://www-path.eecs.berkeley.edu/FSP/
```

To connect to the FSP project site via **netscape** one would type:

```
netscape http://www-path.eecs.berkeley.edu/FSP/
```

Once you have connected to the FSP homepage you will be able to download all of the results, the software programs, and whatever data set you like. The FSP homepage is shown in Figure 12. It is strongly recommend that researchers use this to download the data simply because it is so user-friendly.

8 Conclusion

This paper gave a summary of the by-products of the Freeway Service Patrol Evaluation Project. It described the three different data sets, the problems associated with them, and how they were corrected. It discussed the software developed to process the data, and to correct the mistakes in the data. To correct the loop data it was necessary to first interpolate over the "holes" with the adjacent loop detector data and then to multiply by pre-calculated correction factors. The probe data was used to fix the location and the duration of the incident data. Finally, a way to download the data and the support software has been outlined. The goal is for other researchers to be able to download this data and make use of it.

Acknowledgments: This work was supported by the California Department of Transportation (Caltrans) as part of the PATH Program at the Institute of Transportation Studies, University of California, Berkeley. We would like to thank Mr. Joe Palen of Caltrans Division of New

Technology who served as the contract monitor and provided guidance and assistance throughout the study.

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 views or policies of the State of California. This report does not constitute a standard, specification, or regulation.

References

[1] D.B. Fambro, C.L. Dudek, C.J. Messer, (1976) "Cost-Effectiveness of Freeway Courtesy Patrols in Houston," *Transportation Research Record*, vol. 601, pp. 1–7.

[2] J. A. Lindley, (1986) "Qualification of Urban Freeway Congestion and Analysis of Remedial Measures," Tech. Rep. Report RD/87-052, FHWA, Washington, D.C..

[3] M. Morris, W. Lee, (1994) "A Survey of Efforts to Evaluate Freeway Service Patrols," (Washington, D.C.), 73rd Annual Meeting of the Transportation Research Board.

[4] J. K. Ousterhout, (1993) *Tcl and the Tk Toolkit*. Addison-Wesley.

[5] K. Petty, (1995) *FSP 1.1: The Analysis Software for the FSP Project*. University of California, Berkeley.

[6] Kumud Sanwal, (1994) "An Extended Macroscopic Model for Traffic Flow," *Transportation Research: Part B*.

[7] A. Skabardonis, H. Noeimi, K. Petty, D. Rydzewski, P. P. Varaiya, H. Al-Deek, (1994) "Freeway Service Patrols Evaluation," PATH Research Report UCB-ITS-PRR-95-5, Institute of Transportation Studies, University of California, Berkeley.

[8] S.D. Wohlschlager, K.N. Balke, (1992) "Incident Response and Clearance in the State of Texas: Case Studies of Four Motorist Assistance Patrols," Tech. Rep. Report FHWA/TX-92/1232-15, Texas Transportation Institute, Texas A&M University.

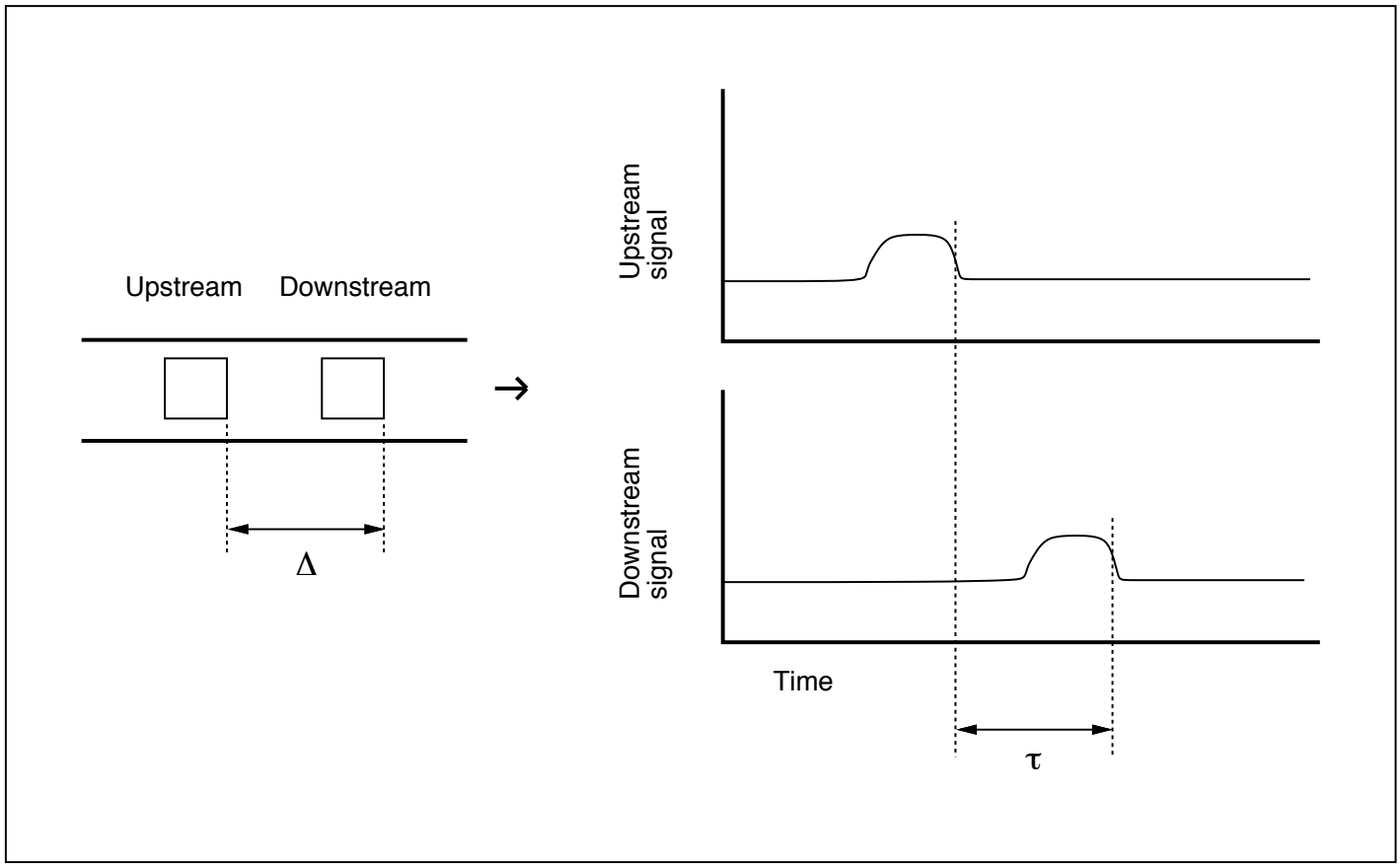

Figure 1: Basic Calculation of Car Speeds.

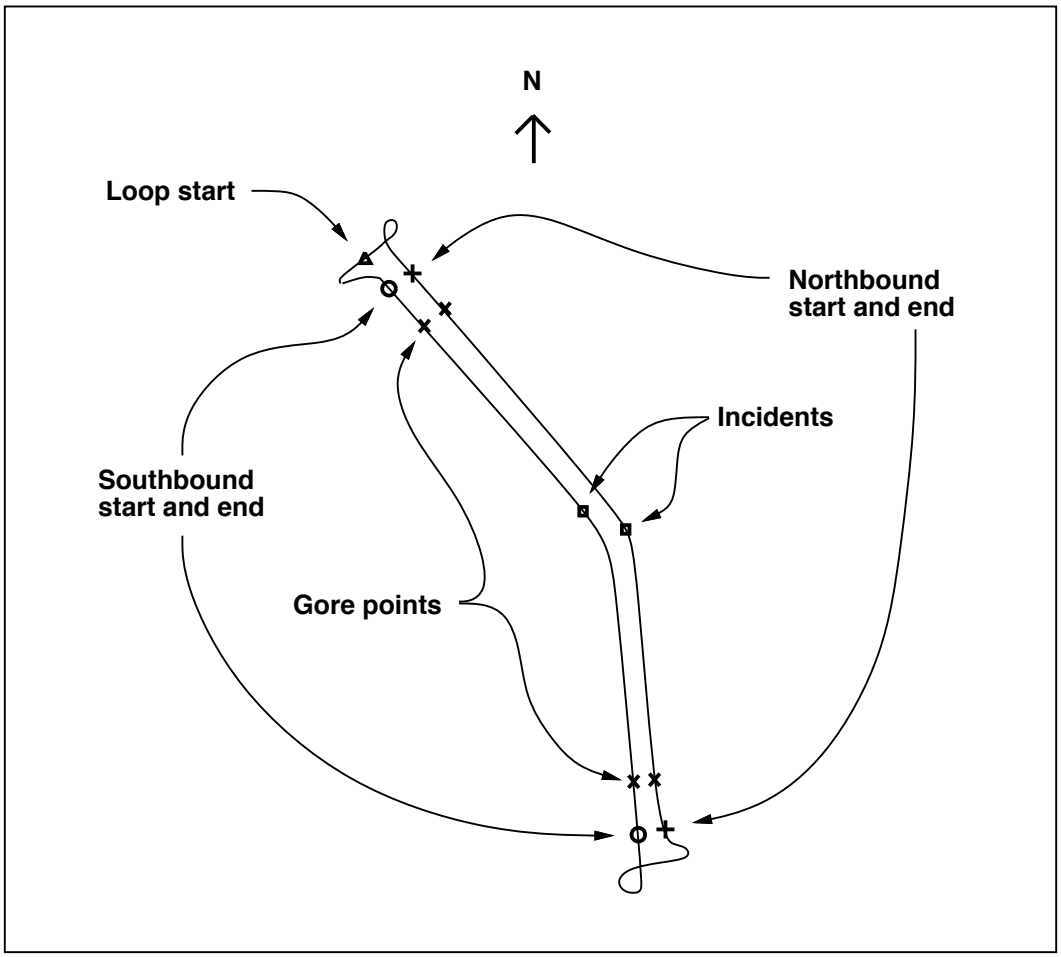

Figure 2: Keys pressed by drivers of probe vehicles.

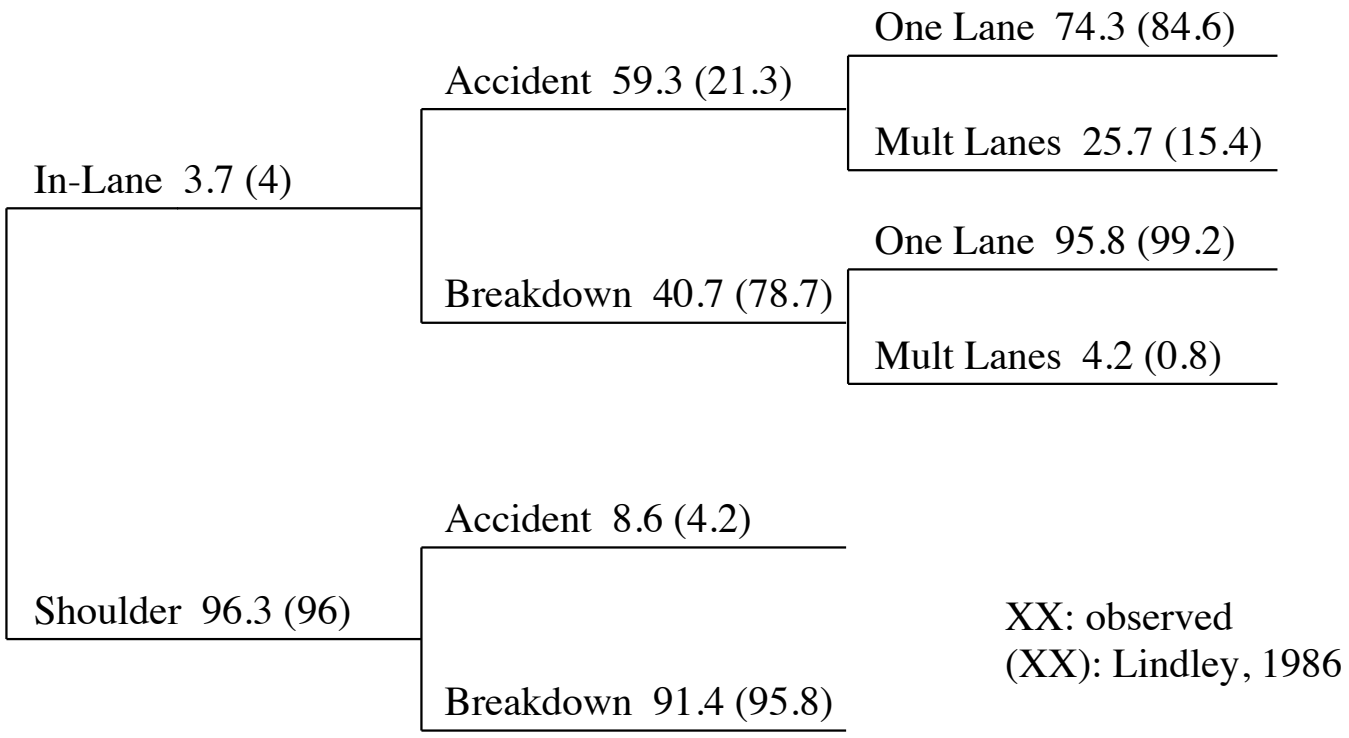

Figure 3: Incident frequency breakdown. Values are in percent.

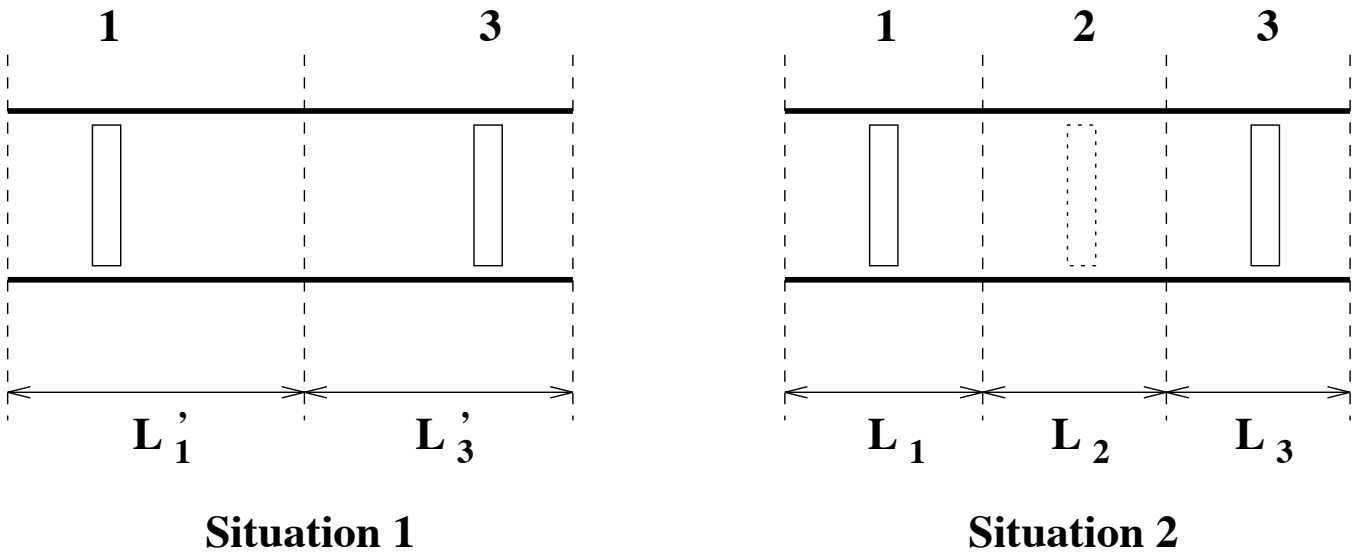

Figure 4: Missing Detector.

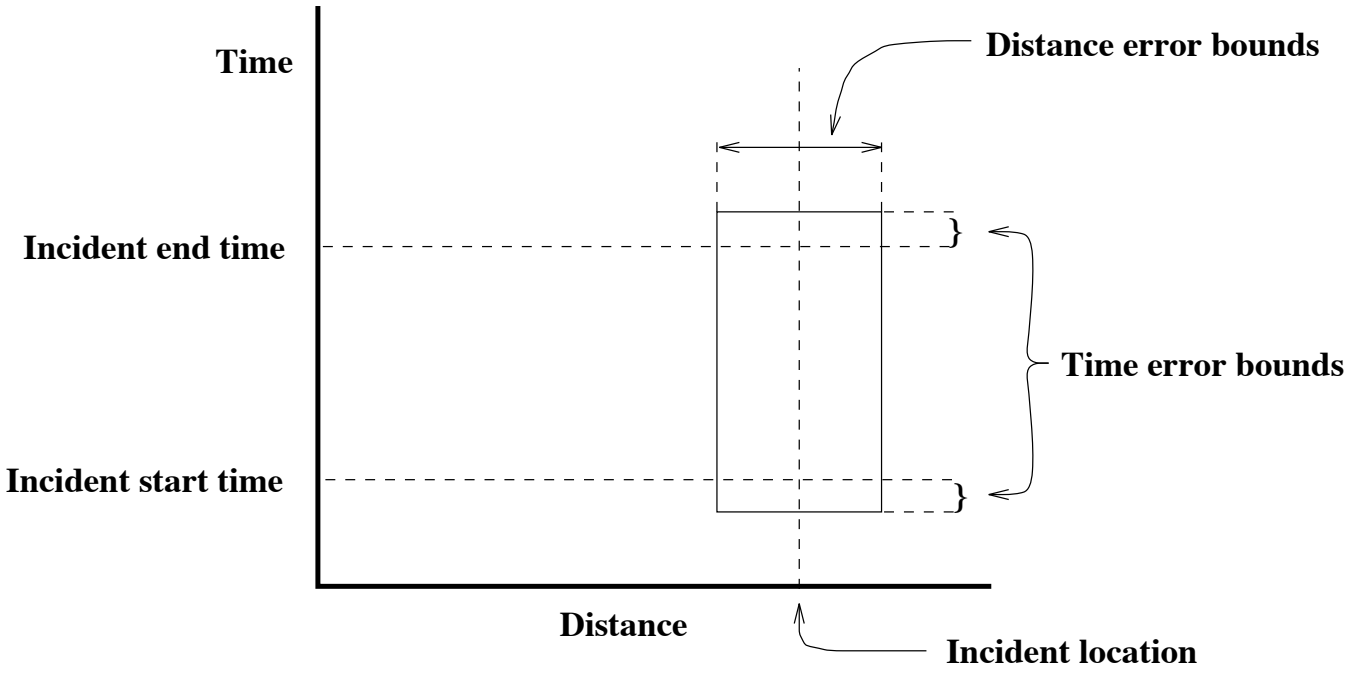

Figure 5: Basic Incident Plot.

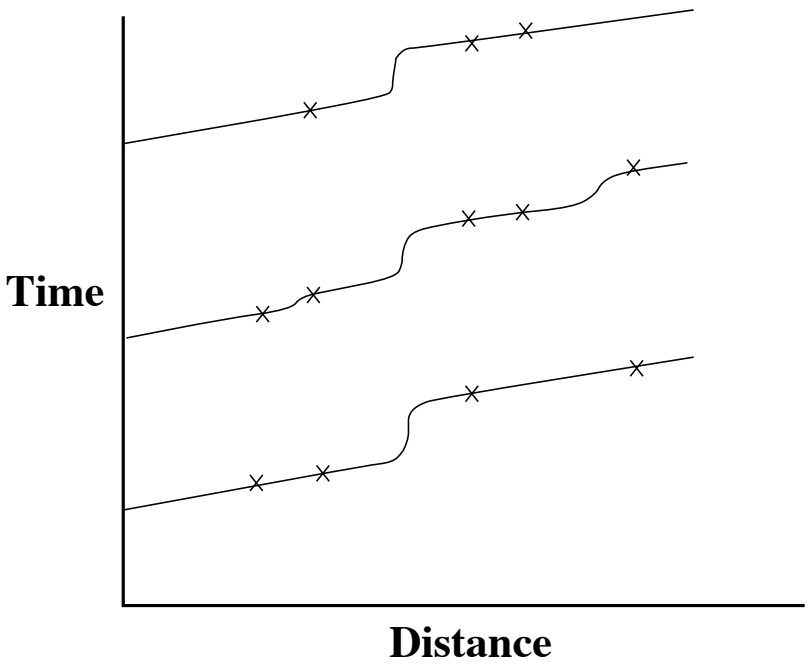

Figure 6: Car Trajectories.

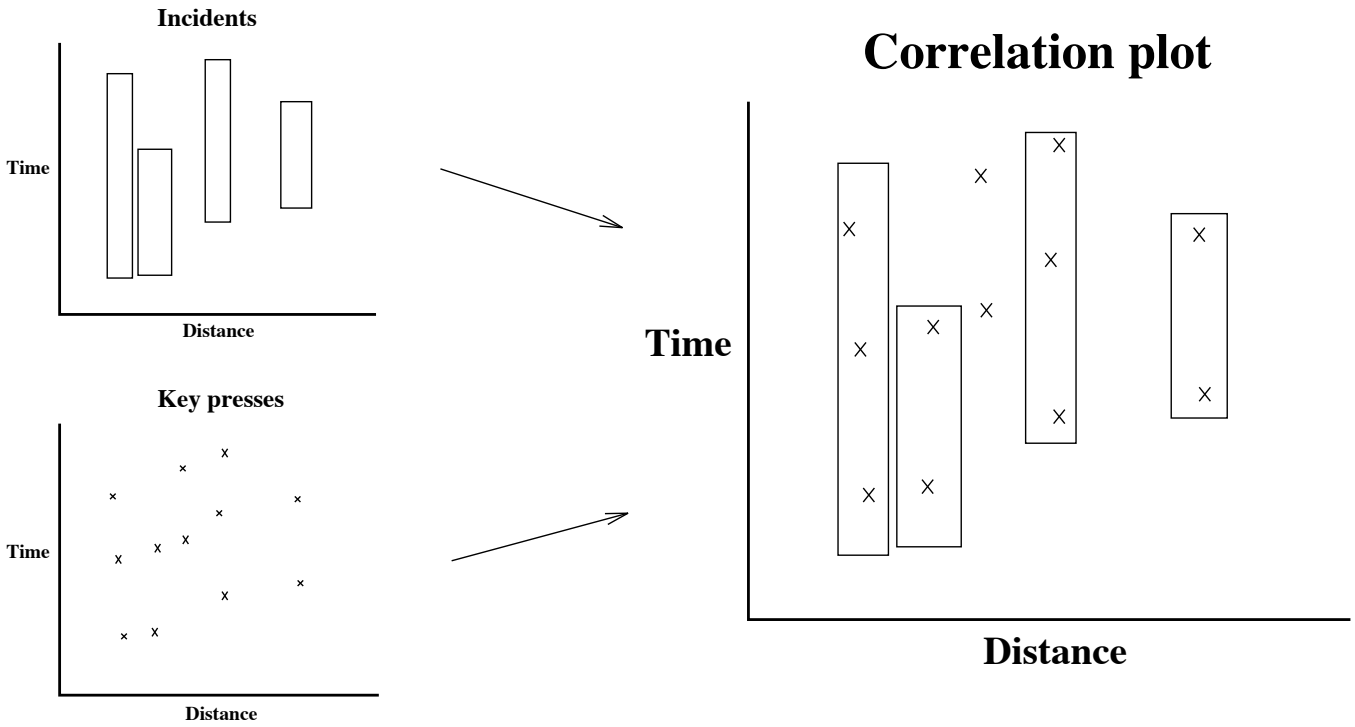

Figure 7: Correlation Plots.

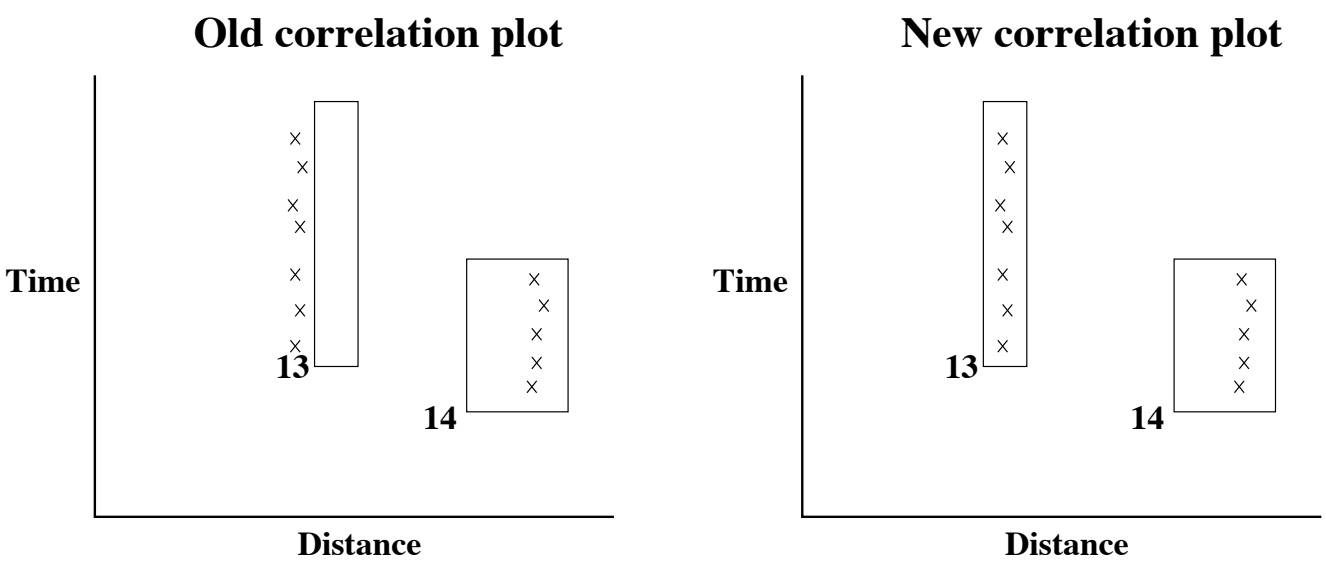

Figure 8: Fixed Incident Placement.

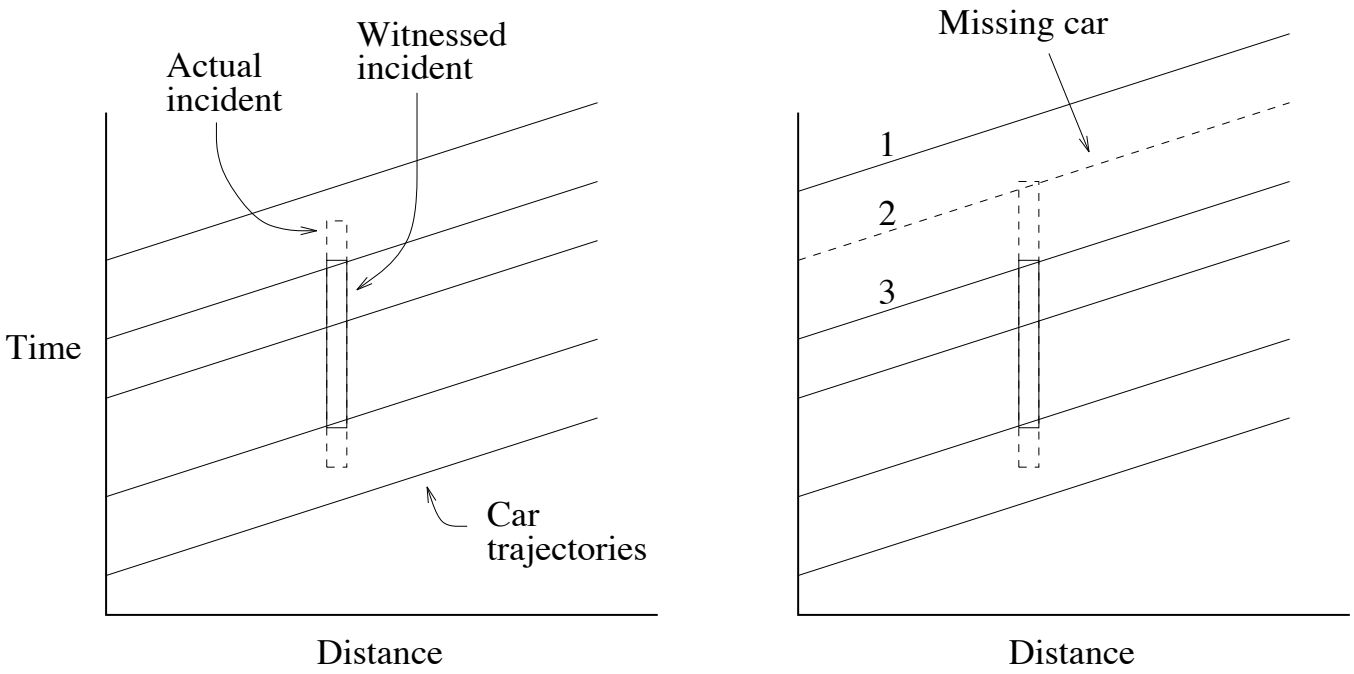

Figure 9: Problems With Fixing Incident Duration.

Data type	Output
Loop Data	Speed vs. Time plots
	Counts vs. Time plots
	Occupancy vs. Time plots
	Delay (v-hr) vs. Time plots
	Text reports of the data
	Error reports on the data
	Reports of dropout times
Probe Vehicle Data	Latitude vs. Longitude plots of car trajectory
	Speed vs. Time plots of car trajectory
	Distance vs. Time plots of car trajectory
	Speed vs. Distance plots of car trajectory
	Link Travel Time vs. Starting Time
	Plots of GPS data
Incident Data	Histograms of incident duration
	Cumulative distribution of incident duration
Data Analysis	Delay per incident
	Plot of delay per incident duration
	Plots of correlation between car and incident data
	Contour plots of traffic delay and density on the freeway

Table 1: Various types of output from the support software.

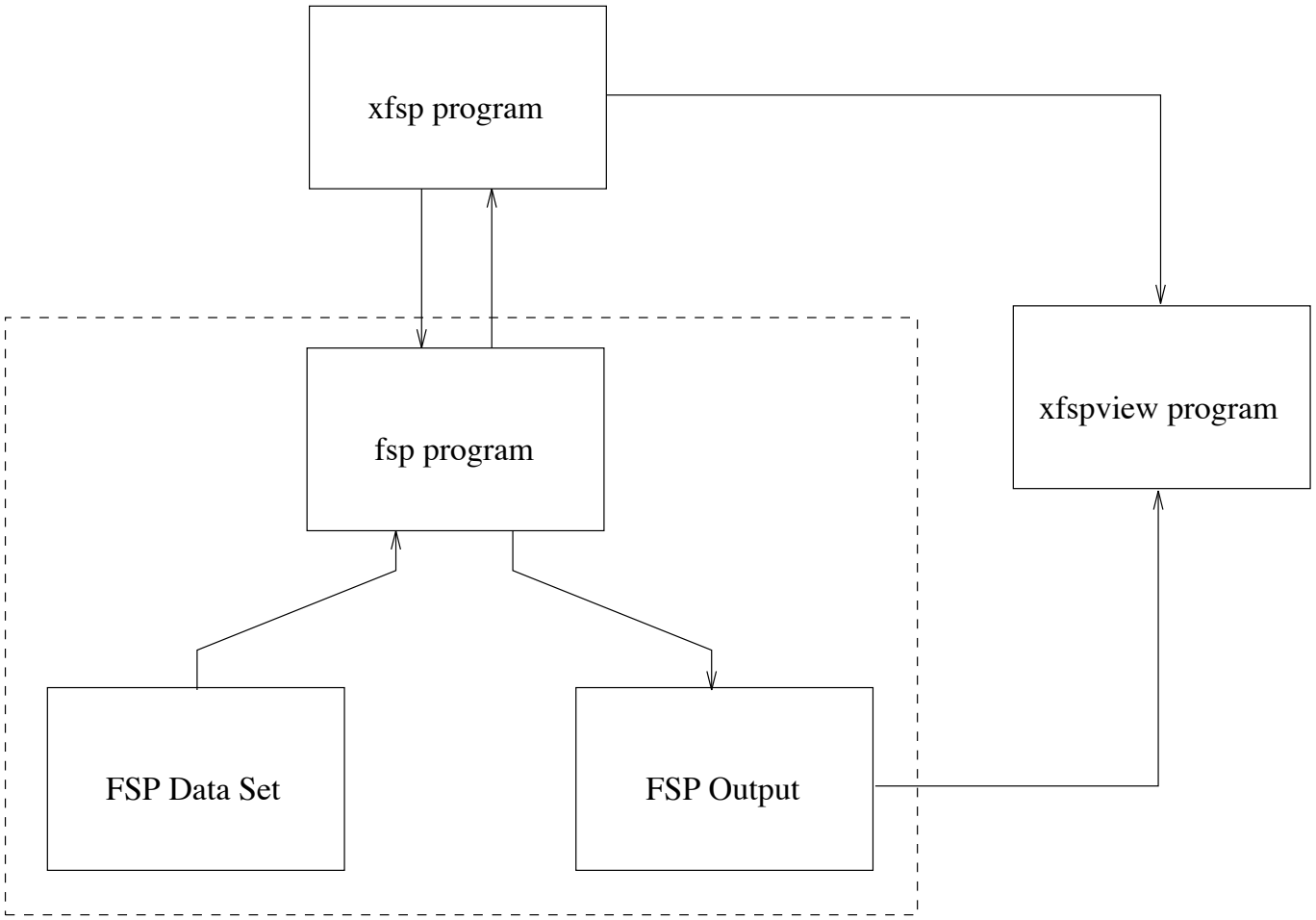

Figure 10: Relationship Between **fsp** and **xfsp** Programs.

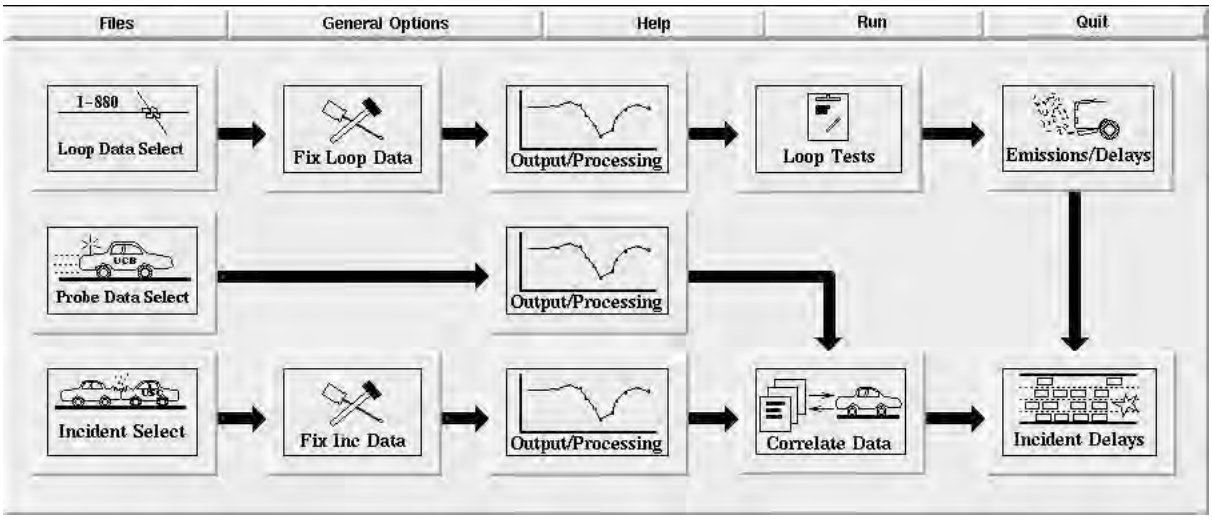

Figure 11: Main **xfsp** Window.

Description	Package path and name
Manual for the support software	/pub/PATH/FSP/Packages/fsp.man.1.1.ps.Z
Support Software	/pub/PATH/FSP/Packages/fsp.1.1.tar.Z /pub/PATH/FSP/Packages/tcl7.3.tar.Z /pub/PATH/FSP/Packages/tk3.6.tar.Z /pub/PATH/FSP/Packages/expect.tar.Z
Raw Data	/pub/PATH/FSP/Data/Raw/Set1 /pub/PATH/FSP/Data/Raw/Set2
Processed Data	/pub/PATH/FSP/Data/Processed/Set1 /pub/PATH/FSP/Data/Processed/Set2

Table 2: Anonymous ftp locations for the software packages.

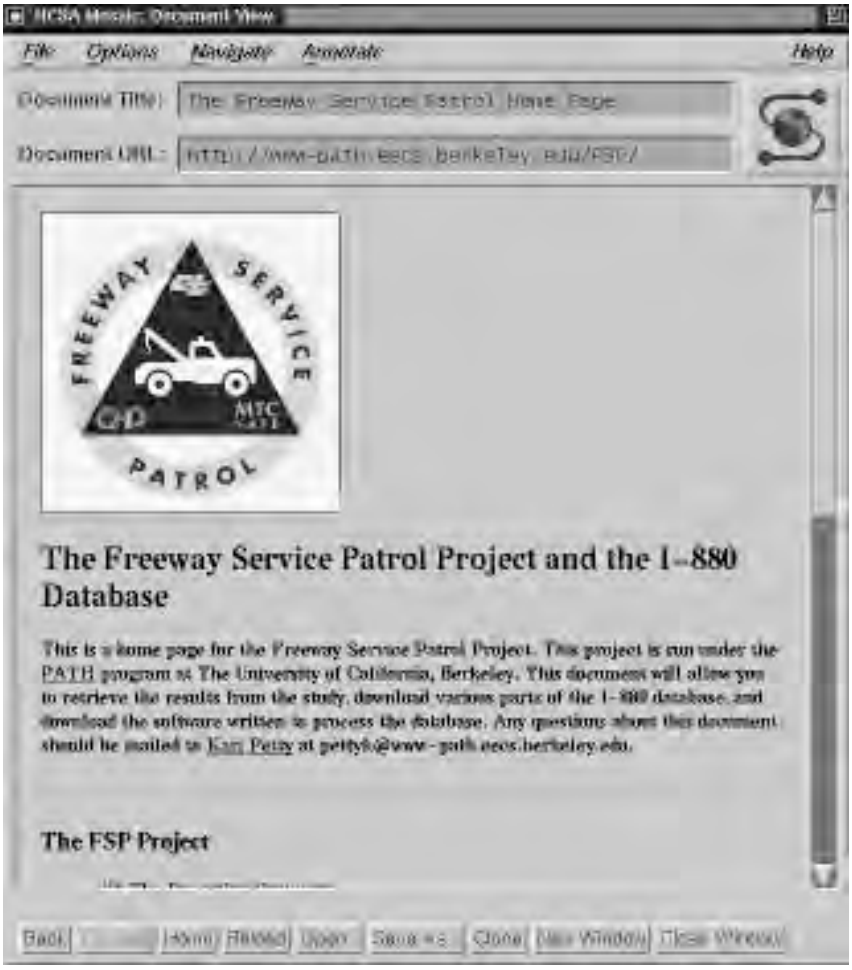

Figure 12: FSP Homepage in Mosaic