A MULTIMEDIA, INTERACTIVE DATA VERIFICATION AND REDUCTION TOOL FOR USE IN DRIVING SIMULATOR RESEARCH

Ginger S. Watson, Ph.D.
Branch Chief, Human Factors
&
Yiannis E. Papelis, Ph.D.
Branch Chief, Simulation Technology
&
Matthew C. Schikore
Senior Systems Programmer
National Advanced Driving Simulator, The University of Iowa
Iowa City, Iowa

Abstract

After executing driving simulator studies, researchers are inundated with data containing primary vehicle state variables such as velocity, Cartesian position, and sensor inputs. These data must be reduced into meaningful quantities that often rely on relative measures against static or dynamic elements of the virtual environment. For example, following distance and time-to-collision depend not only on primary state variables, but also on mathematical relationships between the primary variables and other information. Characterizing driving, assessing human performance, and validating simulator performance require even higher-level data. Because the process for reducing and interpreting driving simulator data is not standardized, it is imperative to be able to visualize the data, both to verify its correctness and to help researchers understand it. Software tools in use and under development at The University of Iowa's National Advanced Driving Simulator integrate online data collection and off-line data reduction and verification by providing interactive facilities for multi-modal visualization and manipulation of data using a hybrid of off-the-shelf and custom software. The linkage between binary data and digital video allows researchers not only to visualize the data, but also to quickly and automatically overview video and online written notes in correlated time.

Introduction

Driving simulators are increasingly used for human-centered research, virtual proving ground applications, and training. One reason that simulators are attractive for so many applications is that they can produce data that describe the synthetic environment and the state of the vehicle in minute detail over extended periods and under repeatable conditions. Therefore, all driving simulators must collect and process data. The processing varies according to application, but it is generally referred to as *data reduction*.

In training simulators, data reduction focuses on computing variables that quantify the performance mastery of the driver relative to the performance criteria stated in the instructional objectives for each training unit. Since training applications require instantaneous feedback, data reduction in training simulators is done in real-time when possible. If it cannot be done in real-time, the amount of post-processing is minimized to deliver results promptly after a lesson is completed. A major benefit of online data reduction is that it facilitates advanced instructional technologies such as individualized adaptive testing and performance-based remediation and practice regimes based on a student's need for practice to ensure mastery of specific performance objectives.

In research simulators, data reduction requirements vary widely. When the research is driver-centered, as in most human factors or medical studies, data reduction is more specialized and may vary from study to study. Typical processing includes the derivation of following distance, time-to-collision, and other variables that measure the response or interaction of the driver with multiple entities in the virtual environment.

In virtual proving ground applications, where a virtual prototype is used in lieu of a physical prototype, data reduction often focuses on vehicle performance. Usually, performance data are provided to other off-line engineering tools, which then

Presented at the **IMAGE 2000 Conference**
Scottsdale, Arizona, 10-14 July 2000.

calculate aggregate engineering-based performance measures. Further data analysis is often necessary when performing tasks such as predicting equipment life based on driver inputs to a simulator.

In all simulator uses, data reduction refers to the process by which raw data obtained by the simulator are transformed into primary or secondary variables that can be employed by simulator users. The process by which reduced data are verified for correctness is called data verification.

The work described in this paper is motivated by the need to automate, as much as possible, the data reduction and verification process for high-fidelity simulators used primarily for human subject experimentation and virtual proving ground applications, with special emphasis on the National Advanced Driving Simulator (NADS).

This paper is organized as follows. It begins with an overview of the NADS device, followed by a description of the process by which experiments are constructed and implemented on the NADS. Next is a detailed description of data reduction, with emphasis on requirements and implementation. Finally, tools that support data visualization and verification are described, and a few examples of their usage are provided.

NADS Overview

The NADS, shown in Fig. 1, is a high-fidelity driving simulator. Its nine advanced technology subsystems interact with the driver to create a highly realistic, immersive driving environment (NHTSA, 1993). The subsystems are described here to illustrate the complexity of the simulator environment and the types of data that can be collected during experimental trials.

Fig. 1 National Advanced Driving Simulator.

(1) Image Generator – The Image Generator developed by Evans & Sutherland provides a 360-degree field of view, including rearview mirror images. The visual data include highway traffic control devices (signs and signals), common intersection types (railroad crossings, overpasses, bridge structures, tunnels), three-dimensional objects that vehicles encounter (animals, potholes, concrete joints), high-density multiple-lane traffic interacting with the driver's vehicle, and roadway weather environments.

(2) Motion System – The motion system provides a 64-by-64-foot translational motion envelope, the largest ever developed for a driving simulator. A hexapod motion base holds a turning platform capable of 330-degree horizontal rotation. Four high-frequency vibration actuators are placed on the cabs to provide road-roughness cues off each tire.

(3) Vehicle Cab System – Four fully instrumented vehicle cabs are available in the NADS: a Ford Taurus, a Chevrolet Malibu, a Jeep Cherokee, and a Freightliner Century Series Day Cab. Each cab is configured with standard and optional instruments.

(4) Control Feel System – Instrumentation and software models provide a realistic feel of vehicle response to the road and to driver inputs to steering, brakes, clutch, transmission, and throttle. The control feel system can represent automatic and manual control characteristics such as power steering, existing and experimental drivetrains, anti-lock braking systems, and headway control systems.

(5) Vehicle Dynamics – High-fidelity vehicle dynamics software accurately represents vehicle motion and control feel conditions in response to driver actions, road surface conditions, and vehicle aerodynamics. Models simulate light passenger cars and trucks, heavy trucks and buses, and off-road wheeled and tracked vehicles. The models encompass normal driving conditions as well as extreme maneuvers encountered during crash avoidance situations, including spinout and incipient rollover.

(6) Audio System – The sound system coordinates three-dimensional audio sources with the other sensory systems. Audio cues are provided for vehicle operation (e.g., engine, tire, brakes, and wind), various road surfaces, road surface changes due to weather, high-density multiple-lane traffic, and objects encountered during a drive (e.g., potholes, concrete/tar joints).

(7) Scenario System – The scenario control modules provide real-time traffic elements such as passenger cars, motorcycles, trucks, buses, rail-based vehicles, and pedestrians. These elements have autonomous, reactive behaviors and provide a complex, realistic environment for the driver. Specialized runtime agents simulate operation of traffic control devices, weather effects including wind

and fog, and ambient traffic. Graphical tools allow users to develop these scenes and scenarios prior to data collection without explicit programming. These tools are described in the experiment development flow section of this paper.

(8) Computer System – All computers that control NADS operations are integrated to provide the visual driving environment, vehicle characteristics, roadway properties, and audio cues that make up the virtual environment. This system allows the user to define and monitor highly controlled experiments.

(9) Simulator Development Module (SDM) – The SDM is a NADS-compatible, fixed-base simulator used for cost-effective development and testing of subsystems prior to full-scale testing on the NADS.

A more complete description of these nine major subsystems can be found in the *National Advanced Driving Simulator (NADS) Functional Specification Document* (NHTSA, 1993).

In addition to the nine major subsystems, researchers can use several tools to define and test the technical parameters of their experiments prior to integration and testing on the SDM or NADS. The experimental definition process is integral to understanding how an experiment is defined and how data are processed.

Experiment Development Flow

Understanding the experiment development flow is important for understanding the data reduction specification process. The experiment specification process includes the identification of the synthetic environment and scenarios, and data reduction is closely linked to these components. The tools used to develop the synthetic environment and scenarios also support data reduction specification and implementation.

Figure 2 illustrates the process by which an experiment is designed for the NADS. The process is driven by the experimenter, who uses several interactive graphical tools to design most of the synthetic environment. A detailed description of the process and all data reduction and verification tools is beyond the scope of this paper. However, a short description of these tools will help describe how they work with the rest of the system.

The Tile Mosaic Tool (TMT) is typically used first. It constructs the static elements of the synthetic environment using tile elements. A tile is a completed rectangular area of the visual database that has been constructed and checked for visual errors or similar anomalies (Papelis & Bahauddin, 1998). Tiles vary in size. A small tile may include a few hundred feet of a residential road and all houses adjacent to the road. A larger tile may include a whole city block, and an even larger tile could represent a small town.

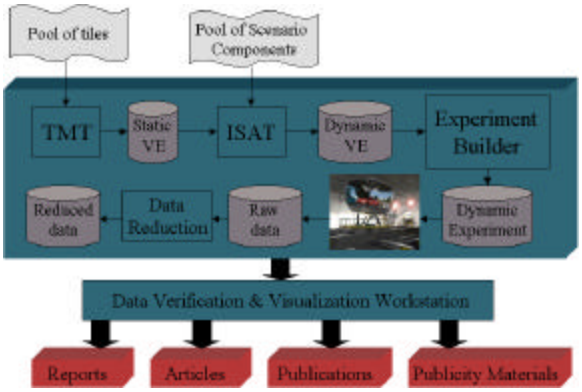

Fig. 2 Experiment development flow.

The TMT allows the user to choose tiles and combine them to create a database to be used with the NADS. It is equipped with features that ensure that the final database has no visual or logical anomalies. For example, the tool will prevent a user from putting tiles whose edges don't match next to each other. The TMT tool is delivered with the NADS.

Once the TMT has been used to construct the static elements of the synthetic environment, the Interactive Scenario Authoring Tool (ISAT) can be used to specify the dynamic elements of the environment. These elements include variables such as the traffic that will be part of the simulation and the environmental conditions and their evolution. For example, a user can specify that the traffic density around the simulator driver be approximately 20 vehicles per mile and that the environment be cloudy with low visibility and wind at 20 mph, gusting to 25 mph. The user can specify the composition of the traffic (e.g., 20% trucks, 60% passenger vehicles, and 20% sport-utility vehicles), and can even control specific behaviors of these vehicles, such as their aggressiveness. In addition, the ISAT can be used to set up special situations such as rear-end collision avoidance scenarios by using techniques that ensure repeatable scenarios that are largely independent of variations in subject behavior.

The ISAT is closely integrated with the NADS scenario control software, a set of programs that can simulate driver behaviors at a very high fidelity. At runtime, the scenario control software reads ISAT output files that specify traffic parameters and driver behaviors.

A key feature of the ISAT is a top-down view of the synthetic environment created by the TMT. It can be used not only to set up the simulator scenarios, but also to help specify required data reduction. In addition, it can be used to specify different data collection parameters dynamically within an

experimental run. Finally, the ISAT can rehearse scenarios by animating the simulator's virtual environment on a workstation. Figure 3 is a snapshot of the ISAT. Like the TMT, the ISAT is delivered with the NADS.

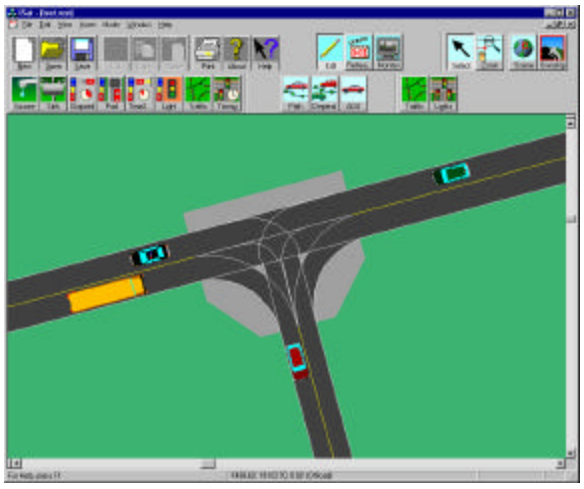

Fig. 3 Interactive Scenario Authoring Tool.

The experiment builder tool is important for data reduction and analysis because it collects and stores all information about an experiment. During an experiment, the information in the experiment builder database can be augmented so that a full audit trail of all activities can be easily retrieved after data collection. The experiment builder collects and stores information such as the number of drivers that are participating in a study and, for each driver, the conditions of exposure on the simulator. In addition, the experiment builder stores the names and dates of data collection files captured on the simulator, along with notes recorded during the drive by the experimenter.

The user of the experiment builder uses subject labels to refer to the participants in a simulator study. Subject labels are arbitrary identifiers associated with a given study that are used instead of the name of the person who drove the simulator. This ensures confidentiality so that there is no way for an experimenter to associate a subject label with the person who participated in a study. The subject labels enable the software to identify a participant and bind together all information obtained during his or her participation in the study.

Once an experiment is completed, participant labels, scenario files, log files, data collection files, questionnaire data, and other information collected electronically during the experiment can be extracted from the experiment builder database. This information can then be fed to the data reduction tools described in the remainder of this document. Like the tools described earlier, the experiment builder tool is delivered with the NADS.

Data Collection

Before data can be processed, they must be collected. Data collection is usually done in real-time by a program that samples variables and writes a record of them, time-stamped with the exact time of collection. Then, at the end of the simulation, a data file containing a series of time-stamped data records is available for processing. Generally, records are sampled at regular intervals (e.g., 10 to 100 times per second) and contain the same data at each time step. The NADS can dynamically change the contents of the data records during collection. The ISAT can create triggers that halt, restart, or change the record layout used by the data collection subsystem. Triggers issue these commands by evaluating various dynamic conditions. Although this capability may minimize the size of the collected data, it does not conceptually alter the data handling process.

Data Reduction

This paper differentiates between *data reduction* and *data analysis*. This is an important distinction because the tools presented here focus on data reduction rather than on data analysis. While the tasks are inherently different, they are often confused as one, leading to poorly designed simulators.

Data reduction is the process of deriving meaningful performance measures from raw data streams. Examples of typical performance measures are average velocity, variance in lane position, mean following distance, or reaction time. Data analysis is the statistical analysis of the reduced variables to determine how performance varies between experimental treatments or conditions. An example of data analysis results is that participants crossed the centerline significantly more often after taking a sedating antihistamine than after taking a non-sedating antihistamine or a placebo (Weiler, et al., 2000).

As another example, consider a research study on cell phone usage where the primary endpoint is the number of collisions between the simulator driver and other vehicles. Any processing performed with the goal of converting the simulator output data into the number of collisions would fall under data reduction, whereas any processing involving how the number of collisions vary would fall under data analysis. Note that this definition does not preclude a simulator from directly outputting data that can be used for analysis.

In fact, training simulators are often designed this way, primarily because the variables are known beforehand. This is more difficult to define for research simulators, however, because they are used for a variety of applications.

The data reduction tools described in this paper were built to simplify the data reduction process. One goal of the NADS was to build a simulator that is flexible and can be used for a variety of research studies, the scope and nature of which cannot be predicted with certainty (NHTSA, 1993). The sheer volume of data produced by the NADS can be invaluable as long as tools exist to help researchers manipulate and understand these data. Although these tools were developed for the NADS, after working with prototypes it has become apparent that they are also applicable to instrumented vehicles. This paper, however, focuses on driving simulator applications.

Based on past requests for data reduction, a two-phase approach to data reduction has been developed. The first phase is segmentation, the process by which the continuous stream of data is broken into substreams. The second phase is the variable generation process, which performs the calculations of actual performance variables within each substream. Both phases are important, first because they are inter-related, and second, because they can significantly simplify the logistics of managing a typical massive volume of data.

Segmentation

Segmentation is the process by which the continuous stream of data produced by the simulator is broken down in multiple substreams, typically representing similar driving maneuvers required over time within the experimental drive. For example, data collected for 35 minutes could be broken down in two substreams—one for the data from the beginning to the end of the fifth minute, and another for the data from the sixth minute to the end of the run. Segments do not have to be contiguous, and do not have to cover the whole data stream (Watson, 1998; Weiler et al., 2000; Romano & Watson, 1994).

Segmentation has two main advantages. First, it can save time when research calls for reduction and analysis only during specific segments. For example, if 500 bytes of data are collected 60 times a second, a 35-minute run would generate a raw data file of over 60 megabytes. Calculating complicated variables over the entire run would be inefficient when variables could be calculated only for the segment needed.

The second advantage is that different data reduction measures can be applied at different segments. For example, a researcher may be interested in how well a subject stays in the lane on a certain curve. Not only would it be inefficient to analyze the entire run in this case, but a measure of lane keeping over the entire run would not necessarily reflect poor performance in the curve of interest.

In order to perform segmentation, one must identify the specific points that mark the boundaries of each segment—the *start mark* and the *end mark*.

There are many ways to determine the start and end marks, depending on the requirements of the experiment and the complexity of the data. One option is to use an absolute time measured from the start of the simulation. This would be useful if a researcher wants to compare how the subject drives in the first five minutes to how the subject drives after one hour. However, this method would not be a good choice for analyzing a subject's performance in a specific area of the database.

Another option is to specify the marks by geographical location, i.e., specify when the vehicle is in specific regions in the synthetic environment. This method would be useful if a researcher wants to analyze performance on a particular type of roadway such as a straight segment or a curve.

Finally, marks can be determined based on arbitrary conditions. For example, the start mark could be determined when the subject's velocity exceeds X, and the end mark could be determined when it drops below Y. Or, if a researcher wants to analyze a subject's ability to follow a vehicle, the start mark could be when the subject's position is within X meters of the target vehicle, and the end mark could be Y minutes later or at the end of road Z. This is a more general method, and, to some degree, it is a super-set of the other methods. It is more complicated, but is by far the most powerful and flexible approach.

Each method has advantages and disadvantages, and each may yield a different number of segments for each subject because different subjects behave differently. With the marks defined by absolute time, a subject might drive very fast and finish the course before one of the segments was to begin. With the marks defined by geographic location, the subject could take an alternate route and not drive over the start mark. If a start mark is defined by when the subject's velocity exceeds X and the subject never does exceed X, that mark will never be set.

The segmentation and data reduction software are capable of dealing with a different number of segments for each subject—important when the process is fully automated.

NADS tools allow users to select criteria for segmentation using any of these methods. Absolute times can be specified for marks, or marks can be specified by geographic location using a top-down view of the database. For more complicated arbitrary

conditions, some coding will be necessary. The user can use LabVIEW to visually code the conditions that must be met for each mark.

Each segment will be assigned a name for the data reduction program to use to identify which variables are computed in each segment.

Each of these tools will calculate the frame number for the start mark and end mark for each segment for each subject. These frame numbers will be output to a file for the data reduction program to read. The data reduction program will be able to read a list of segments with start and end frames for each subject, and will not have to calculate where the segments start and end.

Variable Generation

The data reduction tool responsible for generating variables consists of two layers: the user interface layer and the processing layer.

The user interface layer provides dialog boxes that can be used to select segments and, for each segment, the type of processing to be performed and specification for how the output data will be formatted. A key feature of this tool is that it provides the user with context-sensitive options that depend on all available information about the experiment to this point. For example, it has access to the database produced by the experiment builder and all the data produced by the segmentation phase. The user can select segments either by name or by grouping them for each simulator participant, exposure date, scenario treatment, or other criteria. Furthermore, the tool knows what variables mean and can reject processing choices if they are illogical or if the necessary data are not available. For example, the tool will not allow a segment of data to be scheduled for lane deviation unless the position of the simulator driver appears in the list of available variables.

The processing layer is a library of data processing algorithms that have been coded and tested for correctness. These algorithms are designed under the assumption that the input will consist of a series of data records representing sampled quantities. Any of the columns of input data can be used for the calculations. As stated earlier, the user interface layer safeguards against erroneous use of data. The processing layer contains the code to perform the calculations.

Operations can be performed within a column of data or can be performed on multiple columns of data. Operations for single columns of data include finding the min/max, averages, simple counters for discrete variables, pulse detectors, peak detectors, zero crossing detectors, etc. Examples of operations that span multiple columns include following distance between two vehicles, time-to-collision, and number of lane incursions.

The implementation of the processing algorithms can be done in many forms. The processing layer of the data reduction software uses a documented protocol for transmitting data between the segments and the processing modules. Any number of programming languages or commercial tools such as LabVIEW (1998) can be used to generate the appropriate executable code. One advantage of using commercial tools is the standardization of calculations and the fact that other researchers can re-create the same results in other simulators. If the library of available processing algorithms is not sufficient to cover the needs of a particular study, then additional modules can be developed. The initial library of processing algorithms is expected to expand as more researchers use the tools.

The output of the data reduction algorithms follows the same format as the input data. For certain processing algorithms, the output is actually a series of markers that may be used to further segment the original data. It is also possible to use the output of one processing algorithm as input to another, thus creating sophisticated data reduction capabilities. Furthermore, any number of post-processing steps can be applied to that data to make it ready for input into statistical packages such as SAS or SPSS. Finally, the standardized format of the data makes it possible for the data verification and visualization software to intelligently display it to the user.

Data Verification & Visualization

Data verification as described here involves a researcher checking the accuracy of the output from various data reduction algorithms. Data verification is necessary for two reasons. First, it is difficult to exhaustively test software for accuracy, especially software that is relatively new and has not been used extensively. Second, even when mature software is used, occasional projects are subject to strict regulatory requirements that all computer-calculated data be verified for accuracy.

To address the first concern, coding errors, it is useful to review the potential outcome of erroneous software. Generally, coding errors have one of three outcomes. The program may fail with an exception, it may produce erroneous results that are easy to capture because they are dissimilar to other data of the same type, or it may produce erroneous results that are similar to other data of the same type. Of these, the first is by far the easiest to detect and correct. The second is harder to detect, primarily because there is no apparent program termination. However, it usually does not take long for a

researcher to detect completely unreasonable numbers in the output. Such errors are also amenable to automatic detection by software that checks for numbers within a reasonable range. The third is by far the hardest failure to deal with because the erroneous data can easily be mistaken for correct data, and there is no way to automatically detect it.

The data verification approach used by NADS depends heavily on a set of tools that provides extensive visualization capabilities for both the raw and reduced data. The prototype of the tools described here was developed to help detect erroneous data produced by the third failure described here. Consequently, it focuses on the ability to exhaustively verify each piece of output, if necessary. However, it has become apparent that, in addition to helping with verification, these tools provide a unique perspective into meaning of data. In effect, these tools provide the researcher with a unique understanding that would otherwise not be available. For example, researchers have found it extremely useful to view animations of the simulator vehicle during the period of lane excursions because it is easier to associate the duration of the excursion with the actual severity of the excursion in the simulator's environment. In addition, by observing the videotape of the subject's face during these excursions, it was apparent that many subjects were actually falling asleep. This provided a rationale for the high number of excursions under certain treatments (Weiler et al., 2000).

The overwhelmingly positive feedback of researchers led to the development of more specialized visualization capabilities that provide correlated views of all electronic data collected in the course of a simulator experiment. Correlation is important because, due to the volume of data that is available, it can be extremely time-consuming to manually search and identify relevant pieces of information. The primary goal of these tools, beyond detection of errors, is to help researchers better understand the data, ultimately improving their ability to interpret the numeric results of the reduced data and results of data analysis.

The typical setup of a data verification and visualization workstation consists of many components. Not all components have to be available, and the requirements depend heavily on the amount and format of collected data. What follows is a brief description of the system components and how they interact with each other in achieving the goals described earlier:

(1) Binary data vault – This component coordinates the storage of all data involved in an experiment. Even when using relatively high-density media such as CD-ROMs, it is cumbersome to carry or maintain the data on a system. In addition, data accessibility is often an issue because of privacy concerns. The binary data vault is a networked server that contains the data for all experiments. The server runs customized software that manages distribution of files by experiment, without having to expose the storage details of the various data files. Workstations with the required security credentials can contact the server and gain access to all data. They can either use it directly on the server or temporarily download it to the local system for processing. The data vault is more than a file server because it provides per-experiment authentication and responds to queries regarding data files with only experiment and subject labels as inputs. Typically, the server utilizes several high-speed networking ports with a throughput that often exceeds the speed of CD-ROMs.

(2) Primary PC – This system is a PC workstation that runs Windows NT and hosts the primary data reduction application and can potentially host the ISAT. All plots, top-down map views, and processing operations take place on this PC. Because a large number of windows often must be open at the same time, a PC supporting dual monitors is often used in this role.

(3) 3D visualization PC – This is a PC system that is equipped to display 3D images of the synthetic environment using 3D-capable graphic cards. In conjunction with the ISAT, this PC can display a 3D view of the synthetic environment at a given point in the simulation. The view is similar to what the simulator driver would have observed. The user can control the viewpoint independently, which allows closer inspection of a particular scene in 3D.

(4) VCR stack – Any number of VCRs can be used to replay videotape data that were collected during the simulation. For the prototype of the system, digital VCRs controllable though an RS-232 port were used. The data visualization software can then control the VCRs, a capability that allows all VCRs to automatically, precisely, and rapidly seek to a specific point in the simulation.

(5) VCR monitors – One or more monitors for displaying the output of the VCRs are provided. In cases where the number of recorded tapes is too large, an electronic switch is used to multiplex all VCRs into a fewer number of monitors.

(6) Speakers and headphones – When audio is critical in the interpretation of the data, a set of speakers provides sound reproduction.

Figure 4 illustrates a mockup of the prototype system. The lower sets of monitors are driven by the primary PC, and the monitors on top are driven by the VCRs. Three digital VCRs are shown on the left.

Typically, several such workstations can be made available to researchers, each customized to the specific needs of a researcher or an experiment.

Fig. 4 Data verification & visualization workstation.

Example Usage

This section illustrates how these tools can be used by an experimenter after an experiment has taken place. Keep in mind that, although an initial prototype has been completed, the software is still under development. Not all functionality described here is available at the time of this writing.

The majority of the data-handling capabilities are provided by the primary PC that runs the data reduction application. Upon starting the application, a researcher provides the name of the experiment to be manipulated. The local software contacts the data vault to access the various files associated with that experiment. A full description of the files, their availability, and their size is downloaded to the local system. From that point on, when files are needed, they are automatically downloaded from the data vault and remain on the local system until the user exits the program. Then the local files either can be erased or remain in the system for future processing.

The capabilities of the software at this point are completely context-driven and depend on the degree to which processing has taken place. For example, if data reduction has not taken place, no output data will be available for viewing or verification. However, any of the raw data can be selected and plotted onscreen or onto the printer. In addition, the segmentation menu is available. The experimenter can select an option and perform the segmentation. The segmentation output then becomes part of the experiment and is uploaded to the data vault.

If segments are available, the experimenter can then pick processing algorithms for a segment. For example, let us assume that the experimenter selected a time-to-collision calculation and a peak detector using the velocity as input. Processing can now take place. The time it takes to perform the processing can vary greatly, but it is rarely interactive. Generally, it takes several minutes to complete.

Once processing is finished, the data verification window can be used to pull any of the available variables, which now include the peaks of the velocity and time-to-collision. The user can select any of the velocity peaks and use them as markers. The experimenter can then guide all VCRs to seek to the specified point in the simulation. If a tape has not been inserted in the VCRs, the program will prompt the experimenter to insert the appropriately labeled tape and then seek to the specified position. The experimenter can request a 3D view of the simulation at that time. The 3D visualization PC would then initiate rendering of the synthetic environment with a view port that is centered at the position that the simulator driver was occupying at the time described by the marker. Any other vehicles participating in the scene will also be placed in their respective positions. At that point, the experimenter can independently vary the view port of the 3D display.

Note that in the above scenario a few assumptions were made that must be true for a particular experiment to support this functionality. Recorded videotapes should support time marks and should be labeled with unique labels that have been inserted in the experiment database by the experiment builder. Finally, the positions of the remaining entities in the virtual environment should be part of the data collection output in order for the 3D viewer to display them.

Conclusions

This paper described the specification for a set of tools addressing the data reduction and verification problem for high-fidelity driving simulators. The tools are currently focused on the NADS, but are motivated by earlier work on high-fidelity simulation. These tools are currently under development by The University of Iowa. A spiral model of development is used, with the expectation of two to three prototypes before full functionality can be realized. An initial prototype has been developed and was used to analyze and visualize data earlier this year. Currently, a second prototype is under development. The second prototype is closely integrated with the actual NADS tools and, once completed, will incorporate segmentation and a larger number of processing algorithms than the first prototype. The current estimate for completion of the second prototype is the first quarter of 2001.

Author Biographies

Dr. Watson received her B.S. and M.S. degrees from Southern Illinois University in Carbondale, IL, and her Ph.D. degree from The University of Iowa, where she has been active in simulation research for the past nine years. Dr. Watson received numerous academic awards, including a fellowship from the Link Foundation for Simulation and Training to study the effects of simulator disorientation on performance of drivers with varying levels of experience in different fidelity environments. Dr. Watson's research interests are in the areas of training, operator performance, driver impairment, and simulator validation, fidelity, and disorientation.

Dr. Yiannis Papelis received a B.S. degree from Southern Illinois University at Carbondale, a M.S. degree from Purdue University, and a Ph.D. degree from The University of Iowa, all in Electrical and Computer Engineering. He has been involved in driving simulation research for the past 10 years. He has participated in numerous research and development projects involving driving simulators in the US, Europe, and Asia. Dr. Papelis's research interests include autonomous behavior modeling and synthetic environment modeling and simulation.

Mr. Matt Schikore carried out his undergraduate work at The University of Iowa, where he has been active in developing simulation technology and real-time systems for the past six years. Mr. Schikore's research interests are in the areas of visualization, graphical user interfaces, and real-time simulation technology.

References

National Highway Traffic Safety Administration (NHTSA). (1993). *National Advanced Driving Simulator (NADS) Functional Specification Document.* Washington, DC: Author.

National Instruments Corp. (1998). *LabVIEW V5.1 User's Manual*. Austin, TX: Author.

Papelis, Y., and Bahauddin, S. (1998, August). *Rapid Development of Domain Specific Correlated Databases Using Parametrized Tiles.* Paper presented at the 1998 IMAGE Conference, Scottsdale, AZ.

Romano, R., and Watson, G.S. (1994). *Assessment of the Capabilities of the Iowa Driving Simulator.* (Internal report, Turner-Fairbank Highway Research Center, 6300 Georgetown Pike, McLean, VA 22101.)

Watson, G.S. (1998). Simulator sickness adaptation in a high-fidelity driving simulator as a function of scenario intensity and motion cueing (Doctoral dissertation, The University of Iowa, 1998). *Dissertation Abstracts International, 60-02A*, 0398.

Weiler, J.M., Bloomfield, J.R., Woodworth, G.G., Grant, A.R., Layton, T.A., Brown, T.L., McKenzie, D.R., Baker, T.W., Watson, G.S. (2000). Effects of Fexofenadine, Diphenhydramine, and Alcohol on Driving Performance. *Annals of Internal Medicine, 132*(5), 354-363.