Embedded Data: A Step Forward in the Data Reduction Process

Sunil Bulusu
Yiannis Papelis
Bin Chen
Matt Schikore
Omar Ahmad

NADS & Simulation Center
The University of Iowa
2401 Oakdale Blvd.
Iowa City, IA 52242-5003
(319) 335-4786
Fax : 335-4658
Contact e-mail : sbulusu@nads-sc.uiowa.edu

A key problem with data collected during experimental studies is the expeditious reduction and delivery of reduced data. It is not unusual to generate over 500 MB of raw data for a 20-minute drive on the National Advanced Driving Simulator. Managing, transporting, and reducing the simulator data is a time-consuming process that traditionally starts at the end of data collection. During pilot studies, which are conducted to test the feasibility of an experimental design, speedy reduction of data and prompt statistical analysis is a crucial step in designing the full experiment. In such cases, the time-consuming process of reducing large data files impedes progress and escalates the cost of studies. In order to meet the cost and time requirement of studies, an alternative concept of embedded data reduction is used. Embedded data reduction uses scenario authoring elements that embed information in the raw data stream to calculate performance variables. During experimental runs, this information is permanently embedded in the large data files collected. When importing the data files, performance measures embedded in the data streams are recognized and interpreted, and the reduced data is generated. This paper covers the design and analysis of NADS Embedded Data Reduction and its effect on the experimental design process and scenario authoring. In addition, the paper also includes a brief comparison to traditional post-processing techniques and some of the downsides of embedded data processing.

Introduction

Running experimental studies is the primary goal of the National Advanced Driving Simulator (NADS) and Simulation Center at The University of Iowa. A typical experimental study exposes participants to scenarios that consist of a sequence of events. The events are specifically designed to show sensitivity to the performance measures being analyzed. For example, an experiment that involves studying the effects of blood alcohol level, including its effects on driver awareness, may consist of a lane incursion event. During data collection, several participants with varying blood alcohol levels are exposed to the same event. The collected data is then reduced, which, after analysis, shows how participants with increasing levels of blood alcohol react progressively worse to the incursion event.

The collected data contains dynamic information about the participant’s vehicle, autonomous traffic, signals, and signs in the virtual environment. Among other things, this includes the position, orientation, velocity, and acceleration of the participant’s vehicle and surrounding traffic. The data also contains markers that identify the start and end of specific events that experimenters wish to analyze. The data in its raw form will not make much sense to anyone until it is placed in the context of the event being analyzed. This process is referred to as data reduction, and it involves transforming the data into a compilation of variables that can be easily understood and utilized by the experimenter for further data verification and analysis. For example, reduced data from a blood alcohol level study may contain information like the participant’s accelerator and brake reaction times to the lane incursion event.

One approach that avoids any complications in data reduction is to provide a fixed set of reduced data at the end of simulations. Such data may include collisions, average lane deviations, average speed, etc. However, such data is of little use because it lacks any context within which it can be evaluated. As described earlier, each simulator drive involves several events, each of which has specific requirements. The need to segment data and to provide arbitrary programmability and flexibility in specifying the necessary variables eliminates the fixed data output as a viable option.

In the past, the data reduction phase typically begins after the end of data collection and may take anywhere from a few days to several weeks to complete; the amount of time depends on the complexity of the performance measures and involves data handling overhead, and scanning and interpreting the data. The large amount of time needed for data reduction has created a need for an alternative reduction method that produces quicker results albeit with some reduced flexibility.

The next section describes the current data reduction system and our motivation for designing a faster alternative system. We describe the shortcomings and complexities associated with the current system. Next, we describe in detail the new reduction system, called the Embedded Data Reduction System, and how it complements the current data reduction system. This is followed by examples and then comparisons between the two data reduction systems.

Background and Motivation

Understanding the process involved in developing experimental studies at NADS is crucial to understanding the overall flow of the data reduction process. The vital aspects of an experiment include identifying the sequence of traffic events, the road network, and the various performance measures that need to be analyzed. Once the requirements are finalized, traffic events along with reduction measures are authored, tested, integrated, and readied for data collection. On the NADS, data collection is a real-time process during which participants are exposed to various traffic events and their responses, such as brake pedal pressures, steering wheel angles, eye tracker data, etc., are recorded and time stamped. The resulting raw binary data files are referred to as Data Acquisition (DAQ) files. This raw data must be transformed into meaningful performance measures, such as average velocity, reaction times, etc., across different traffic events and then statistically analyzed. The following figure details the data reduction process at NADS.

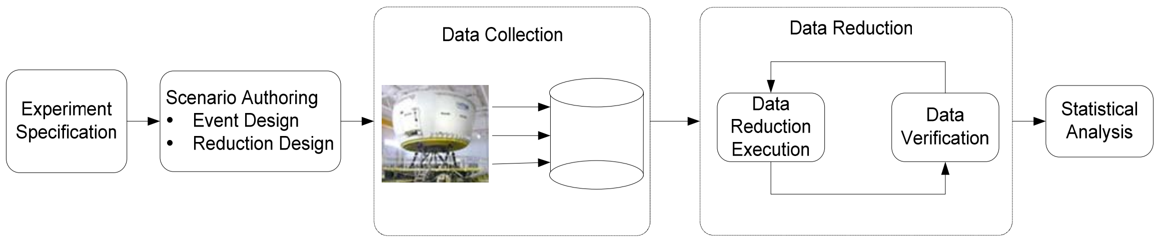


Figure 1: Data reduction process

A multitude of software components and tools have been developed to reduce the DAQ files. These tools include data reduction and verification (DRAV) tools [1] and MATLAB scripts and libraries [9]. Central to the DRAV tools are two components. One of them is a GUI tool that allows users to design dataflow diagrams in order to visually represent the computations required to reduce data [1]. The execution engine, which is the other component, processes these dataflow diagrams and outputs the reduced data. For every experiment, after data collection, time is consumed in designing the specific dataflow diagrams that comply with the study requirements. Any changes to the specifications further delay the reduction process. Refer to the article by Papelis et al. [1] for more details on the design and implementation of DRAV tools. MATLAB is also widely employed as a data reduction tool. Various MATLAB libraries and scripts have been developed to read the DAQ files and reduce data. Again, depending on the reduction specifications, scripts have to be coded and different functions have to be written and executed to derive the performance measures from the DAQ files.

The two methods described above are carried out after the experimental data collection, and significant resources are required to design and execute the dataflow diagrams or to develop MATLAB scripts. The following sections describe some other issues with post-data reduction processes.

The size of the data for a given run of the experiment depends on the number of variables, their data types, the frequency at which they are being collected, and the length of the run or scenario. Variables are collected at 60Hz, 120Hz, or 240Hz, depending on the requirements. For example, a simulator drive lasting 15 minutes with 75 variables being collected at various frequencies results in data sizes of 150-180 MB. A study with 40 participants, with 4 simulator runs for each participant, would have a total size of 30-40 GB of collected data. Transferring a huge volume of collected data results in network overheads. In some studies different participants experience different traffic scenarios. Tracking which scenarios are driven by which participant and applying associated data reduction calculations can be challenging when considering the fact that studies often involve hundreds of participants. Due to the issues discussed, reducing simulator data is a time-intensive process.

Most experimental studies involve running one or more pilot studies to confirm compliance with study requirements and the existence and validity of performance measures. This involves reducing the collected data to make certain that all the information is present to allow the experimenter to complete the analysis. In these situations, fast reduction and delivery of data for compliance becomes a major factor in determining the start of the main part of the study.

Another complexity in performing data reduction is that markers must be inserted in the raw data to indicate the start and stop of each relevant event. This helps correlate the data to the associated event, thus identifying which part of the reduction to apply to each part of the data. This is a step that requires tedious work along with thorough and complete documentation to communicate the details from the person who authored the scenario to the person who will eventually reduce the data. Once the person responsible for reduction receives the documentation, it takes significant effort to actually code the reduction before it can be run on the collected data to produce reduced data for analysis. The overall process is error prone, and provides little opportunity for reuse across studies.

This paper describes a new approach for specifying data reduction activities as part of the scenario in a way that allows a single data reduction program to inspect a data file and swiftly produce the results without any need to know the context within which the data file was collected.

Embedded Data Reduction System

To counter some of the issues with post-data reduction, a new reduction process was developed that embeds data reduction specifications into the scenarios. The scenario author is provided with a detailed scenario specification that describes all of the traffic events the driver will experience. The author uses a graphical scenario authoring tool, the ISAT [3], to implement these traffic events. Events are defined using coordinating agents called triggers, which wait for specific circumstances to occur and then perform actions that affect the scenario in a user-specified way. Embedded data reduction is designed to be a part of the scenario authoring process. So, during authoring of traffic events using the ISAT, embedded data reduction actions are added to the triggers used to initiate traffic events and to triggers that detect the end of an event. These actions have a predefined set of data reduction measures that can be produced. By making use of scenario authoring elements, performance measures specific to the traffic event are embedded in the raw data collected during the simulator runs. Figure 2 details the different phases in the embedded data reduction process, and Table 2 describes some of the measures that are available as embedded data.

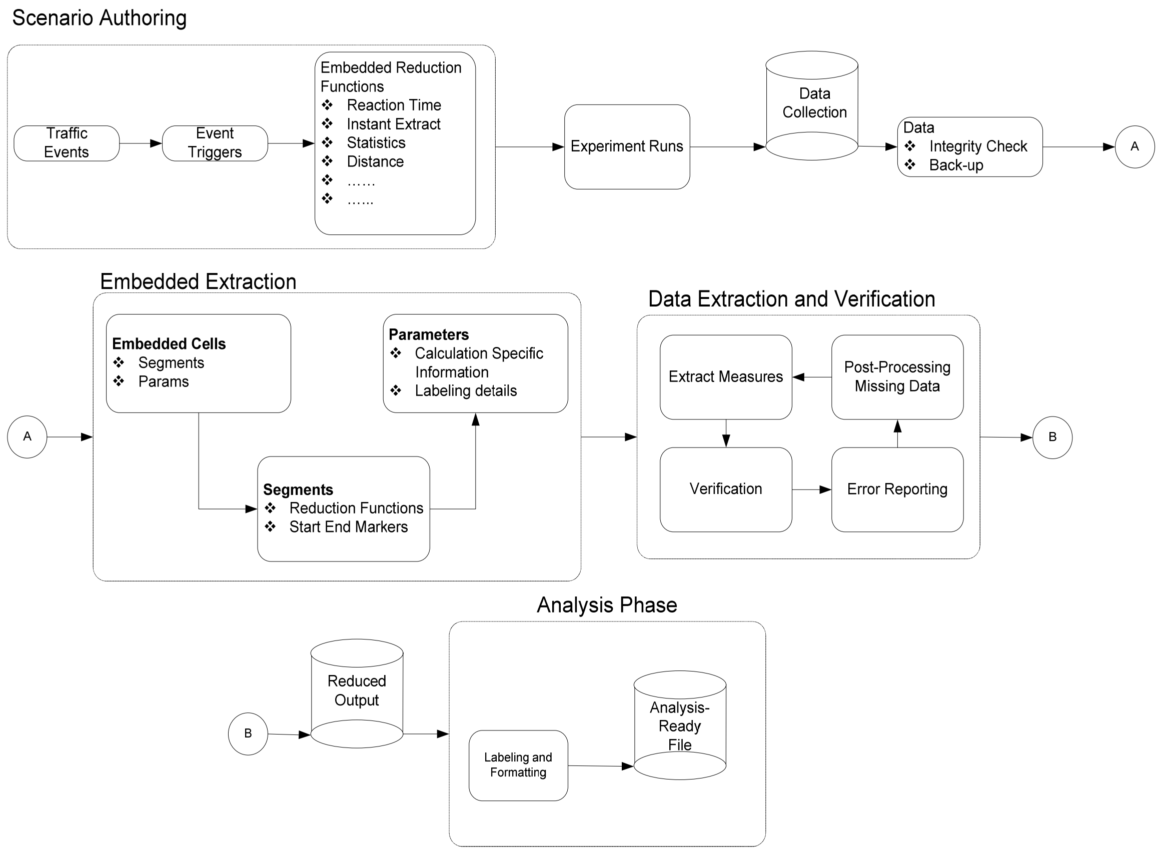


Figure 2: Embedded reduction process

During experimental runs, when a traffic event occurs, the appropriate data reduction triggers are fired and required measures of the traffic events are permanently stored in the cells meant for embedded data. The current embedded reduction process involves two cells: *segments* and *parameters.* Segments have details about the exact reduction functions that need to be applied during a particular event and the start and end timestamps of when data reduction should be performed. Parameters have other calculation and labeling details, which are usually project-specific. This additional information collected by parameters helps to identify the variables collected and eventually gets used in the labeling and formatting of the analysis-ready file.

Since embedded data is part of the cells, it is directly stored in the DAQ files. After every run, DAQ files are backed up and integrity checks are applied to assess their validity.

During the backup of DAQ files, the system detects the data reduction streams, interprets their contents, and writes out a text file containing the reduced data. The initial text file is comma-separated, can be easily imported into Excel, and is ready for inspection. This text file is verified to ascertain the existence of all the required measures and to report any missing measures. The case of missing data arises due to errors in the authoring of data reduction measures, missing data from the data acquisition subsystem, or issues with the interpretation of the embedded data in DAQ files. On most occasions, missing data can be recovered using post-processing of the raw data. After the initial output file is verified, it is formatted and labeled into an analysis-ready format similar to that of SAS Statistical Software [10].

Embedded Data Reduction Calculations

The following table contains some of the data reduction calculations the system offers along with a short description. It is expected that additional measures will be added as necessary.

Table 1: Embedded reduction functions

Name	Description
ReactionTime	Computes the time between the start of a segment and some arbitrary condition, encapsulated in an expression
Maximum	Computes the maximum quantity in a specified cell
Minimum	Computes the minimum quantity in a specified cell
Average	Computes the average of a specified cell
Standard Deviation	Computes the standard deviation of a specified cell
Distance	Computes the distance traveled by the participant during an event
Instant Extract	Extracts the specified cell data in the instant when the trigger fires the data reduction action

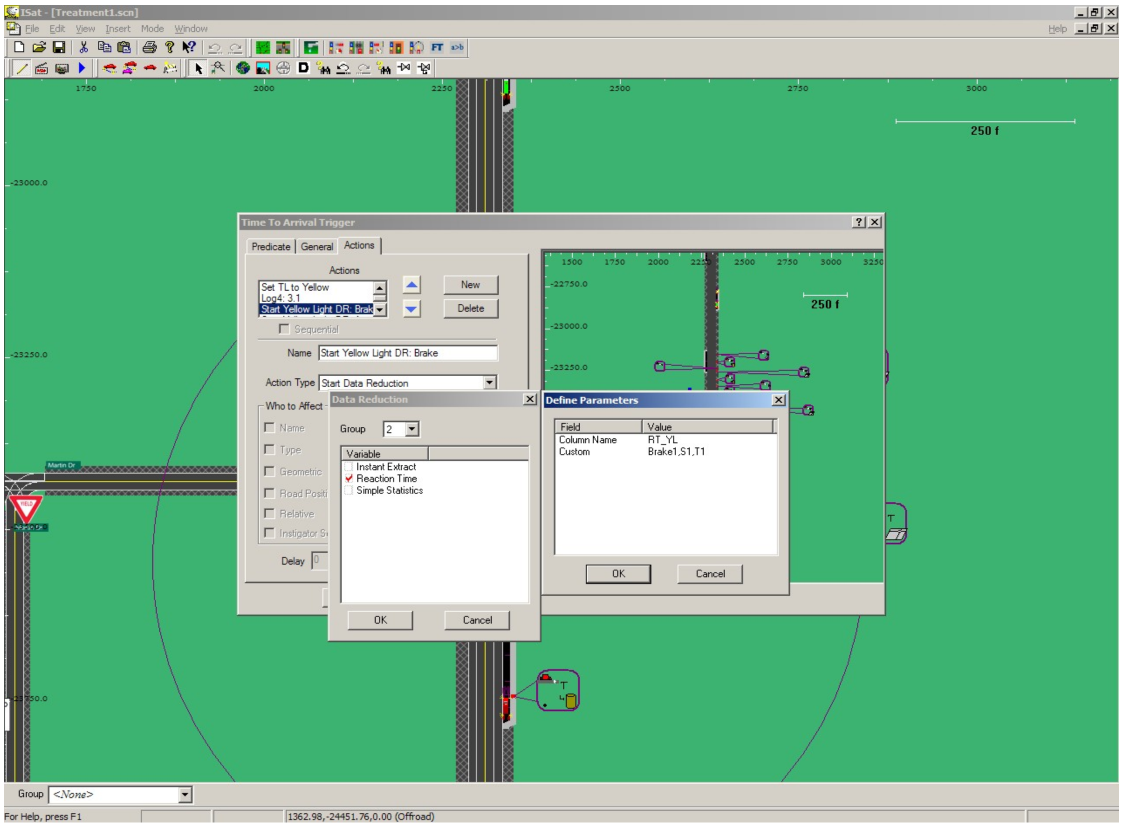
Examples

The following examples describe the process of embedded data reduction and include figures that detail scenario authoring and the final output file. Let us assume that the following traffic event is part of a study. The driver is cruising on a two-lane highway. The vehicle in front performs an emergency braking maneuver while another vehicle occupying the adjacent lane is preventing the driver from changing lanes. This event is used to study the driver's reaction to the braking of the lead vehicle. Two measures are set for embedded data reduction.

The first measure is to calculate the reaction time from when the driver sees the brake lights of the lead vehicle to when the driver applies the brakes. Therefore, a brake pedal press reaction time is the time elapsed between the display of brake lights in the lead vehicle and when the brake pedal is pressed by the driver. The second measure is used to study how close the driver is to the lead vehicle when brakes are applied. Headway is the time needed for the driver to reach the current location of the lead vehicle.

These measures are authored as two segments. The first segment, Segment A, is used for the brake pedal press reaction time. When the lead vehicle's brake lights are turned on, a marker indicating the start of Segment A will be authored. This marker includes a measure identifier and other parameters essential for calculating reaction time. When the brake pedal is pressed, another marker indicating the stop of Segment A is authored. Figure 3 shows the details of authoring the brake pedal press reaction time. As soon as the data reduction trigger is activated, the start marker for reaction time will be recorded in the data stream. The second segment, Segment B, is used for headway. The start marker is set as soon as the driver applies the brakes. The instant extract marker includes the measure identifier and the name of the cell that collects headway details. Figure 3 shows the details of authoring instant extraction of headway.

At runtime, the embedded data reduction cells are used to store the data from Segment A and Segment B. After the data are collected, an embedded data reduction program is used to extract embedded data reduction information from the cells. From the examples described above, it will find two segments, Segment A and Segment B. In Segment A, it will find the start frame and end frame, the measure identifier, and other calculation parameters. The reaction time will be calculated from start and end frames and, along with the reaction time, the output would also include the parameters specified. From Segment B, it will find the start marker, the measure identifier, and the cell name (Ex: SCC_Follow_Info) for instant extraction. The output in this case consists of the value in the cell at that instant. Table 2 shows the final output from the embedded data reduction.



Time To Arrival Trigger

Predicate | General | Actions

Actions:
- Set TL to Yellow
- Log4: 3.1
- Start Yellow Light DR: Brak

Sequential

Name: Start Yellow Light DR: Brake

Action Type: Start Data Reduction

Who to Affect: Name, Type, Geometric, Road Positi, Relative, Instigator S

Delay: 0

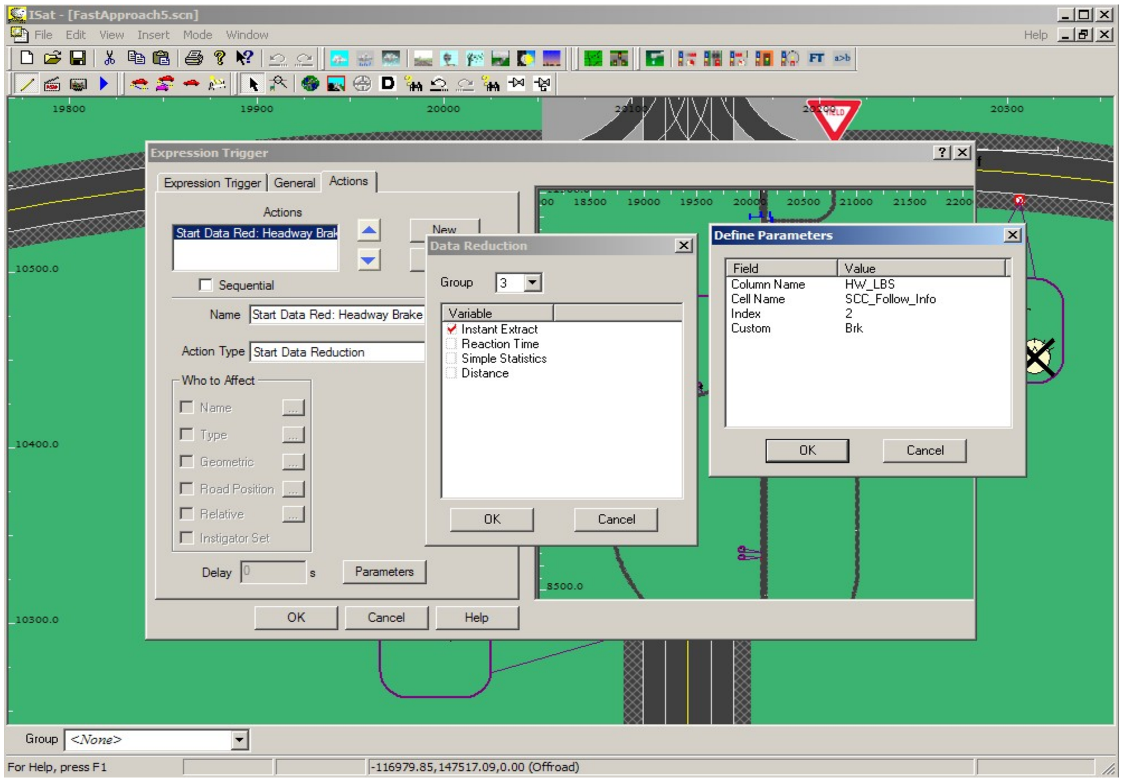
Data Reduction

Group: 2

Variable
Instant Extract
✓ Reaction Time
Simple Statistics

Define Parameters

Field	Value
Column Name	RT_YL
Custom	Brake1,S1,T1


ISat - [FastApproach5.scn]
File Edit Insert View Mode Window Help
19800 19900 20000 20100 20200 20300
10500.0
10400.0
10300.0
Group <None>
For Help, press F1
-116979.85,147517.09,0.00 (Offroad)

Expression Trigger

Expression Trigger | General | Actions

Actions:
- Start Data Red: Headway Brak

Sequential

Name: Start Data Red: Headway Brake

Action Type: Start Data Reduction

Who to Affect: Name, Type, Geometric, Road Position, Relative, Instigator Set

Delay: 0 s

Data Reduction

Group: 3

Variable
✓ Instant Extract
Reaction Time
Simple Statistics
Distance

Define Parameters

Field	Value
Column Name	HW_LBS
Cell Name	SCC_Follow_Info
Index	2
Custom	Brk

Figure 3: Authoring embedded data

Table 2: Embedded data reduction output

Subject	Run	Instance	Reaction	Start Marker	End Marker	RT [secs]	Parameter	Instant Name	Instant Frame	Instant	Parameter1	Parameter2	Parameter3
S01	R01	I01	RT_YL	12300	12540	1.0	Brake1	HW_LBS	12540	1.2	SCC_ Follow_ Info	2	Brk

Comparison

Data reduction by post-processing is a traditional and a time-tested process. Since embedded reduction is a relatively new concept, some of the advantages and disadvantages are presented below.

Advantages

• *Methodology*
The main difference between embedded and post data reduction lies in the methodology. As discussed earlier, embedded reduction is completed with an analysis-ready file at the end of data collection, and post reduction usually begins at the end of data collection. Every time new measures are required, post-processing requires new code to be developed based on the unique requirements of the reduction specifications.

• *Early Specification*
The need to author the data reduction as part of a scenario forces both experimenters and developers to think early about the data requirements. As in any discipline involving complex design, early design reviews are more efficient and create less errors than after-the-fact development.

• *Reusability*
Scenario events, by their nature can be reused across studies with little overhead. Since data reduction is part of the scenario, the specification and the coding for the data reduction is also reused.

• *Standardization*
The availability of complicated measures that are easily reusable promotes standardization between studies.

• *Performance*
With calculations taking place in real time, reduced data can be extracted with little overhead

• *Configuration Management*
The mapping between a scenario and the associated data reduction calculations can be challenging when considering the fact that studies often involve hundreds of participants, each experiencing five to ten scenarios and each scenario involving numerous events. With embedded data reduction, the need to maintain mapping is eliminated because the data is self-contained in the DAQ file.

Limitations

• *Functionality*

Post-processing makes use of the raw binary data that is collected during simulator runs for generating the output files. These binary files have data about all the complex subsystems involved in the functioning of the NADS and have numerous cells recording data about multiple subsystems. Post-processing requires the raw data to have some details about the performance measures, and just about any measure can be coded and derived from raw data. Currently, the scope of embedded reduction is limited to only a finite set of performance measures that can be recorded and reduced.

• *Statistical Comparisons*

An important difference between embedded and post reduction is that embedded data cannot be used to produce measures across different participants or runs. For example, comparing lane excursions across different drivers in a single run cannot be done using embedded data. But as post-processing involves coding and is performed at the end of data collection, almost any measure can be reduced, including comparisons across multiple drivers.

• *Sampling Rate*

From a data collection standpoint, the sampling rate of the data used by post reduction is different from the sampling rate of the data used by embedded reduction. Post-processing makes use of DAQ files in which the sampling rate varies from 60 – 240Hz. Embedded data reduction is taking place as a part of scenario control execution, which runs at 30Hz. The effect of different sampling rates is that sometimes the exact measure when calculated using post data might be different from the one calculated using embedded data; however, when the final output is analyzed, the difference is small enough not to affect the final analysis.

• *Authoring*

Another issue that needs attention is that with embedded data reduction there is an enormous increase in the usage of data reduction triggers on top of existing event triggers. From an authoring standpoint, every measure on a traffic event requires a trigger to initiate data reduction and another to terminate. Termination of data reduction depends on the response to that event, and sometimes there are multiple responses that terminate the data reduction. This leads to multiple triggers tracking the different responses and leads to further crowding of the scenario. This problem is specific to the implementation of the NADS scenario authoring tool and may not be applicable elsewhere.

From the comparisons made above, it can be concluded that there are definite advantages to embedded data processing, as well as some significant deficiencies in the system.

Conclusion

This paper described a new concept of embedded data reduction that is designed to hasten the data reduction process. Although embedded processing does not yet offer the wide range of features and functions available in existing post-processing techniques, it offers many advantages for experimental studies and pilot studies that require limited performance measures. Post-processing is still required for studies that require measures beyond those offered by the current version of the embedded process, but it shows great potential and is definitely a step forward in the data reduction process. It should be strongly pursued and improved upon by adding more functionality.

References

1. Yiannis Papelis, Matthew Schikore, Ginger S. Watson, Tad Gates. A Prototype Toolset for Rapidly Reducing and Reviewing Driving Simulator Data, *Proceedings of the 1st Human-Centered Transportation Simulation Conference* (CD-ROM, ISSN 1538-3288), Iowa City, Iowa, November 2001.
2. Ginger S. Watson, Yiannis Papelis, Matthew Schikore. A Multimedia, Interactive Data Verification and Reduction Tool for use in Driving Simulator Research, *Proceedings of the IMAGE 2000 Conference*, Scottsdale, Arizona, July 10-14, 2000.
3. Matthew Schikore, Yiannis E. Papelis, Ginger Watson. Advanced Tools for Graphical Authoring of Dynamic Virtual Environments at the National Advanced Driving Simulator, *Proceedings of the Driving Simulation Conference*, 2000.
4. Harold Klee, Essam Radwan. Assessment of the Use of a Driving Simulator for Traffic Engineering Studies, Final Report USDOT, Washington DC, 2004
5. Stéphane Espié, F. Saad. Feasibility of the use of a driving simulator for in-depth human driver behavior studies, *Proceedings of the Driving Simulation Conference*, 2000.
6. Andrew H Hoskins. Development and Validation of the Pennsylvania Truck Driving Simulator, Thesis submitted to Pennsylvania State University, 2002.
7. A.H. Jamson, H.C. Pyne and O.M.J. Carsten. Evaluation of traffic calming measures using the Leeds Driving Simulator, *Proceedings of the Driving Simulation Conference*, 1999.
8. S. Millemann, S. Panis, P. Loslever, J.C. Popieul, P. Simon3, M.A. Dillies-Peltier1Car driving activity: Human and vehicle behavior on a long monotonous journey, *Proceedings of the Driving Simulation Conference*, 2001.
9. Matlab User's Guide, The Mathworks Inc, 2001.
10. SAS Statistical Analysis Software, The SAS Institute Inc.