第 13 回計測自動制御学会制御部門大会 (2013)

Development of a driving simulator for vehicle control and ITS research

Yoann PENCREACH[*1]

[*1] Forum 8 Co., Ltd. (www.forum8.co.jp) VR Development Group
Gakuen Kibanadai-nishi 2-1-1, Miyazaki-ken Miyazaki City, 889-2155 Japan

Abstract: In this paper, we discuss how we addressed some of the synchronization issues in HILS system and why solving synchronization can impact on the latency, then how with a simple technique we could improve both latency and synchronization. The studied driving simulator uses a real time vehicle dynamics module connected to our virtual reality application, where the simulation of the environment is FPS dependent.

Key Words: Driving Simulator, ITS, Vehicle control

1 Introduction on testing ECUs of a vehicle in a virtual environment

Virtual reality is a tool widely used for simulation where the interaction with an actual person is necessary. It is used for instance in systems for training on emergency situation, flight and drive simulators are also well known as efficient means to train or test pilots and drivers. But virtual reality is also used for the reverse purpose: test an actual part of a system under real usage conditions. A good example is given by the ECU or Electronic Control Units; as their complexity increases, the use of simulation where the actual ECU hardware is integrated (HILS) becomes necessary.

Since the 80s the number of ECU has incredibly increased in our vehicles in order to improve their efficiency, safety and comfort. Many different types of ECU exit such as the Powertrain Control Module (PCM), Brake Control Module (BCM), Transmission Control Module (TCM), Body Control Module (BCM), Suspension Control Module (SCM), etc… Each of them controlling a specific part of the vehicle according to information acquired through various types of sensors. The number of ECU on a vehicle can exceed 50 nowadays, and the heavy research effort in vehicle makes complexity of the overall system increase rapidly. Programming an ECU has becomes a great challenge for the engineers increasing the need in simulation, validation and experiments.

In 2000 we (Forum 8 Co., Ltd.) released the first version of UC-win/Road[*1] a commercial virtual reality software application. At that time we already had a long expertise in development of software for civil engineers and UC-win/Road started as a tool for consensus and review of road and city planning and designs. UC-win/Road included from its first version an extensive set of tools to model in three dimensions the infrastructures of the roads. From that point the product was extended to allow seamless simulation of the traffic and later driving simulation in the 3D virtual environment.

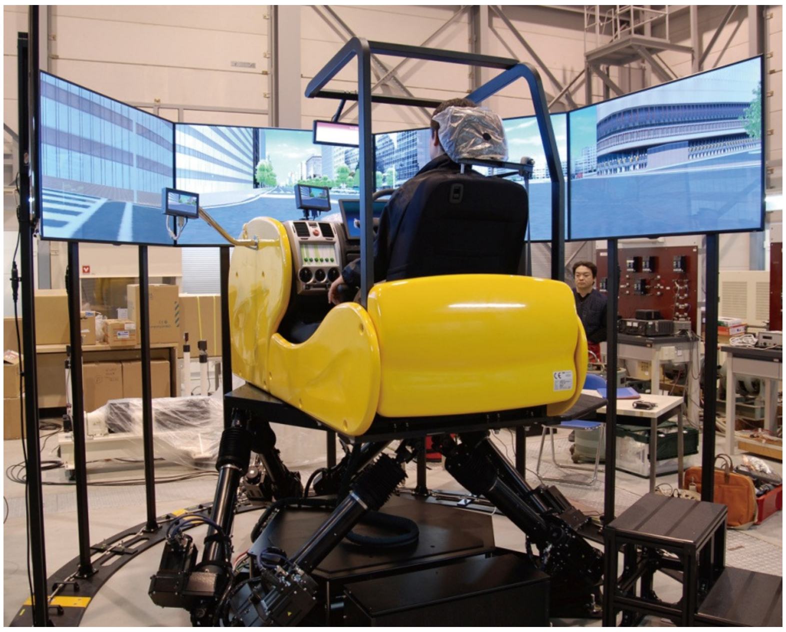


Fig. 1: Example of driving simulator using UC-win/Road.

Since then we participated at many projects involving technologies like HILS and biometry instruments for the driver. As many equipments and modules have to operate together latency and synchronization issues become very important for the comfort of the user but also to allow the hardware to work correctly in the simulation.

As shown in Figure 2 the ECU outputs control signals directed to other parts of the vehicle while the input can come from various sources of information. The most common and the first king of information used in ECU is information coming from other parts of the vehicle itself. But the information can be also about the driver, the road condition, surrounding vehicles or pedestrians. More recently with the increasing interest in Intelligent Transport Systems (ITS) the use of

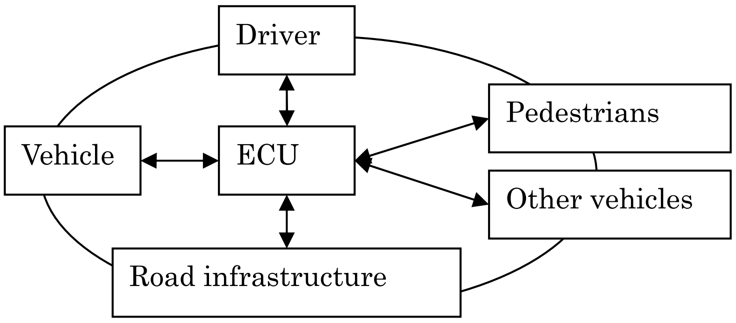


Fig. 2: An ECU and its environment

[*1] UC-win/Road official web site: http://vr.forum8.co.jp/

real-time data transmitted from other vehicles, pedestrians or infrastructure equipments has become also very important.

In order to test and develop an ECU in a simulation environment it is essential to reproduce all the input signals that the ECU will receive in a real environment and also be able to execute many different scenarios that an ECU may encounter in a real driving situation. For that the simulator must be able to simulate the data coming from the vehicle itself, from the driver and the traffic and other aspect of the environment. A standard set of tools would be a software module for the vehicle physical simulation, a virtual reality system connected to a driving simulator so a real person can drive the simulated vehicle and a traffic simulation engine. Each of them should be highly customizable to allow an easy integration of other subsystems needed for the research.

If we consider the entire simulator system, the ECU is not the only part that requires a simulation for its input but the driver also needs his stimuli in order to carry out the experiment. This aspect of the system will not be discussed in detail in this document but is also a subject of great interest covering may fields like traffic simulation, artificial intelligence and behavior modeling, visualization, sound generation, 3D model generation, force feedback and motion cueing etc…

A typical driving simulator system using UC-win/Road and integrated with an ECU can be represented as shown in Figure 3.

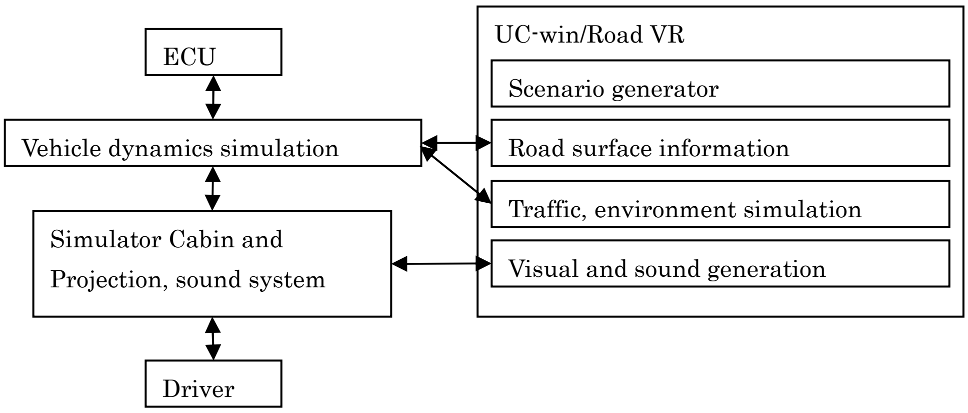

ECU
Vehicle dynamics simulation
Simulator Cabin and Projection, sound system
Driver
UC-win/Road VR
Scenario generator
Road surface information
Traffic, environment simulation
Visual and sound generation

Fig. 3: UC-win/Road in a HILS environment.

This kind of configuration is also called a Hardware In the Loop Simulation (HILS) because a real ECU (hardware) is connected to the simulator and plays its role in the simulation. The main elements are as follow:

(1) ECU connected to the Vehicle Dynamic Simulation (VDS). Usually all the data the ECU needs are sent from the VDS and the ECU can also control back the vehicle simulated in the VDS.

(2) The Simulator cabin and the driver whose role is to control the VDS.

(3) The traffic simulator has two roles: show a realistic traffic condition to the driver and send real time information to the ECU via

the VDS.

(4) The road condition including mainly the road surface geometry and friction coefficient used to feed the VDS with the parameters needed in the vehicle simulation. This can be done offline before starting the simulation or in real time for changing conditions. This information is also used for visualization and is shared between the virtual reality engine and the VDS.

(5) Virtual reality (VR) engine: its role is to provide as much as possible a convicting experience to the driver in terms of visual and audio queue.

2 How to deal with a variable frame rate in a HILS environment?

2.1 Context

In the test environment the ECU must receive the same data, at the same rate as in a real environment. The VDS module is responsible for sending the data of the simulated vehicle in real time to the ECU. In order to ensure a real time simulation and data transfer at a constant rate, a real time operating system (RT OS) is used.

UC-win/Road is connected to the VDS and runs on a standard Microsoft® Windows® operating system. In this environment a constant data transfer cannot be ensured mainly for two reasons:

1. The operating system is not designed to provide a strictly constant timer and also cannot ensure the availability of the resources at a given time for an application.
2. UC-win/Road provides tools for the user to build his driving environment. Therefore the application allows the user to create 3D environment that may not be able to sustain a constant image refresh rate also called frame per second (FPS).

Regarding the real time issue, we could improve the system with techniques such as using material interruption and virtualization to almost force a constant simulation time step. However, because the amount of data to be handled by the visualization is variable and because we want to keep that flexibility in our application the FPS is potentially variable.

In any case the result of the VDS must be somehow synchronized with the actual frame displayed on the screen. We will explain later what would happen without synchronization. In this paper we will study the case where the traffic simulation is calculated in UC-win/Road at a variable time step and the driver's vehicle simulation handled by the VDS at a constant time step.

2.2 Description of our solution

The traffic and environment simulation in UC-win/Road are calculated to follow the actual measured FPS as shown on Figure 4. The simulation runs in a separate thread from the drawing process to improve the overall performances of the system. However before starting the drawing process of a frame both threads are synchronized. The drawing thread waits for the calculation of the current step of the simulation to be completed. As the time interval between steps is not constant, the duration of the time interval to be simulated at a particular step can only be estimated. In UC-win/Road the time interval of one step of the simulation is chosen equal to the actual time interval measured between the two previous frames. This ensures the total time of the simulation is equal to the actual time of the computer.

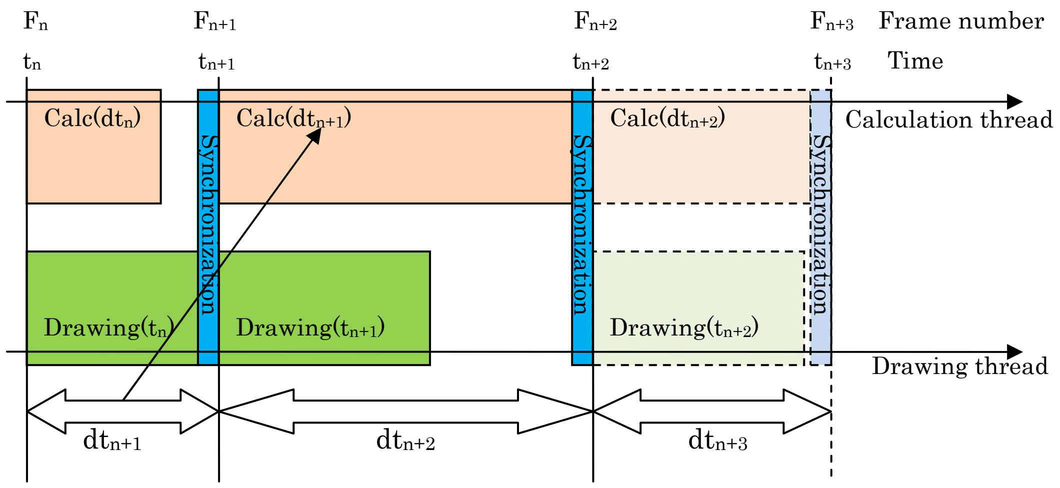


Fig. 4: Simulation and drawing synchronization in UC-win/Road.

This creates a discrepancy as the actual time interval between the current frame (F_n) and the previous frame (F_{n-1}) may not be equal to the interval between the two previous frames (F_{n-1} and F_{n-2}). Between two images separated in time of [t_n - t_{n-1}] seconds the simulation progressed of [t_{n-1} - t_{n-2}] seconds. These two durations can be different because of the varying FPS and this difference could induce a wrong instant speed perception for the user. It would be interesting to study what levels of variation are acceptable for the user; however we will consider for now that except for large fluctuation of the FPS this issue can be ignored.

The next synchronization issue we studied is between the VDS and the VR application.

As shown in Figure 5, if we consider a simple system where the latest information coming from the VDS is integrated in the displayed frame without processing, we realize that the perceived time of the traffic and environment is not the same as the one of the VDS. Although a constant difference may create the impression the traffic does not respond in a realistic reaction time to the actions of the user, this difference in time is also fluctuating as the FPS does. The “wrong” reaction time of the traffic is actually very difficult for the user to perceive when the difference is small and can be adjusted in the traffic algorithm according to the average difference or delay between both simulations. But the fluctuation creates a real discomfort for the driver and prevents from carrying out a driving simulation when a realistic traffic situation is important. Because of this variation, the relative positions between the user's vehicle and other moving objects become unstable. Concretely, the user sees other vehicles as if they were capable of constantly doing sudden decelerations and accelerations.

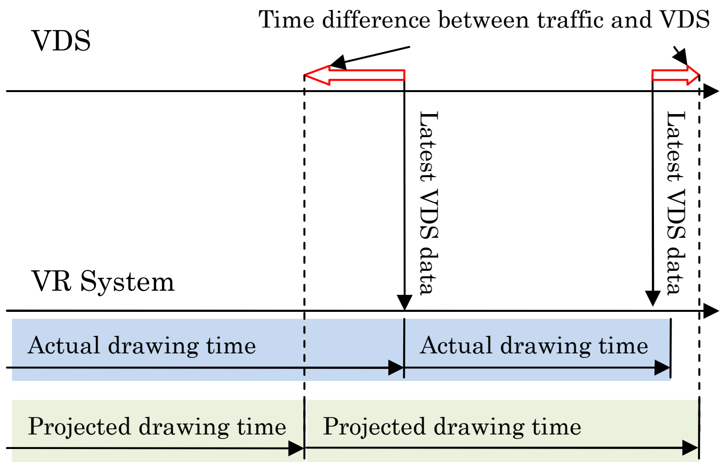
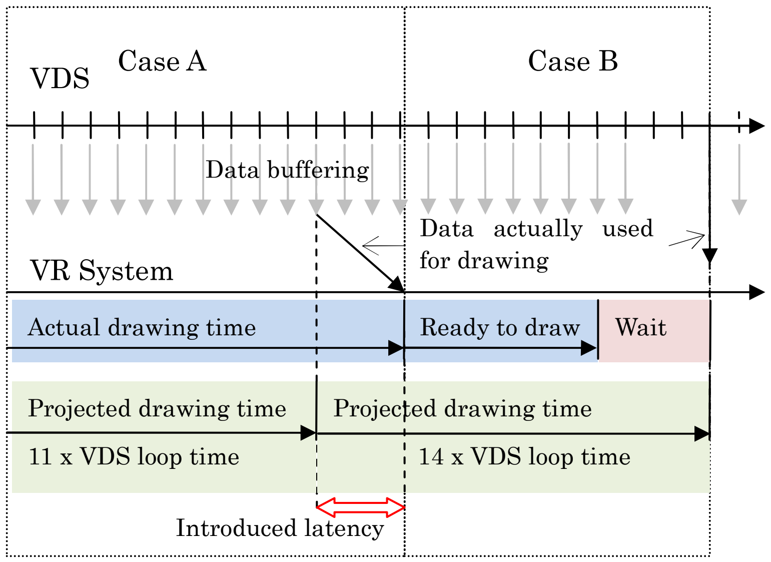


Fig. 5: Basic synchronization issue with HILS

To address that issue we had to improve the estimation method for the time interval in the traffic simulation module. We also implemented a simple buffer used when synchronizing the three processes: 1. traffic and environment simulation, 2. drawing thread and 3. VDS. The first improved mechanism is represented in Figure 6.



The VR system and HILS are synchronized but an undesired latency is introduced

Fig. 6: VDS and visual synchronization at a variable FPS.

However because we cannot control the combined time taken for the traffic simulation and drawing processes, it may differ and be longer or shorter than the projected interval. Yet the calculation is finished and the data from the VDS must be used to represent the user's vehicle in the virtual environment.

In Case A, if the data for the time T has been sent already from the VDS to the VR system and is stored in the buffer, the buffer data is used and older data is deleted from the buffer. Case A occurs when the FPS drops.

In Case B, the drawing thread waits for the VDS to send the data corresponding to the time T. Case B occurs when the FPS increases.

We understand now why the time interval is rounded to the closest multiple of the VDS loop time. If not, the VDS may never be able to send the data for the time T, as T is never simulated.

This first improvement ensures that the data of the user's vehicle and traffic simulation are perfectly synchronized. However we see that in Case A, some undesired latency is introduced in the visual system as we do not use the most recent data coming from the VDS. As long as Case B does not occur the latency will not be reduced to its minimum value.

Typically, after a drop in FPS an additional latency is artificially added for the needs of the synchronization and as long as the FPS is not restored to its original value, the latency stays in the system.

This issue can be simply improved by adding the introduced latency time in the next time interval of the traffic calculation.

In Case B, it is the reverse case, there is no impact on the latency but as we force the visual system to wait for future VDS data, the system runs at lower FPS than its maximum capacity. To resynchronize the systems, the wait time is subtracted from the next time interval of the traffic calculation.

Figure 7 shows the final improved mechanism after a drop in FPS.

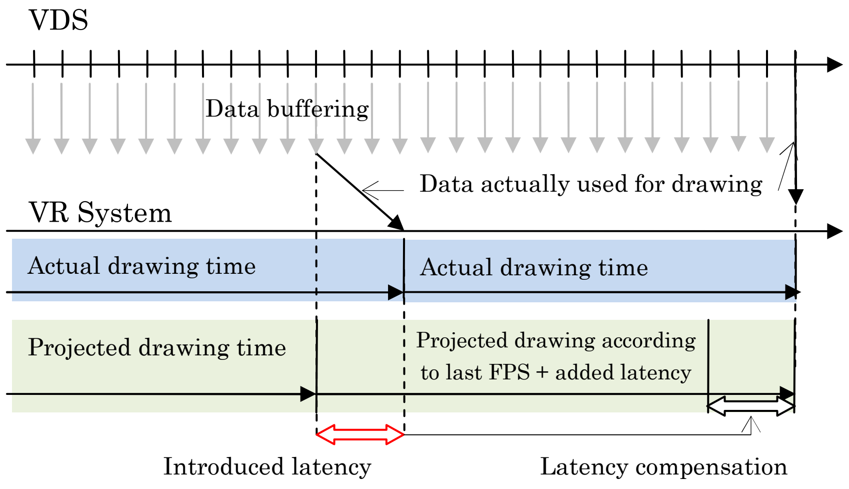


Fig. 7: Latency compensation in VDS and visual synchronization.

3 Conclusions

As the complexity of our vehicle increases, the need of test and experiment increases. To carry out such experiments safely but also as close as possible to a real situation, synchronization and latency issues have to be taken in account in the design of virtual reality applications. In driving simulators various synchronization issues exist, such as the synchronization of data collection from several independent measurement equipment, audio-visual synchronization, motion cues and visual cues etc… This time we saw that while allowing a maximum flexibility for the user about the content of his experiment, it is possible to implement a robust connection between a real time system and visualization at a variable frame rate.

As future development we plan to extend our system to enable low latency synchronization between multiple driving simulators. We believe that it will be an efficient tool for studies on human factors where various driving situations needs to be generated quickly and easily by the researchers and engineers.

References

(1) Forum 8 Co.,Ltd. "UC-win/Road reference manual" ftp://ftp.forum8.co.jp/forum8lib/ucwin/road/UCwinRoadV80000ManualJpn.exe

(2) Michael Meehan (Stanford University), Sharif Razzaque, Frederick P. Brooks, Jr. Mary C. Whitton (UNC-Chapel Hill) "Effect of Latency on Presence in Stressful Virtual Environments"

(3) Rohit Goyal (Ohio State University) "Simulation Experiments with Guaranteed Frame Rate for TCP/IP Traffic" (1997)

(4) Ing. Petr Bouchner (Czech Technical University) "Driving Simulators for HMUI Research"

(5) Chris Schwarz, Yefei He, Andrew Veit (National Advanced Driving Simulator, Iowa) [N11-001] "Eye Tracking in a Cots PC-Based Driving Simulator: Implementation and Applications" (Image 2011 Conference)

(6) Yefei He (National Advanced Driving Simulator, Iowa) [N06-025] "NADS MiniSim Driving Simulator" (Technical Report 2006)