Integrating Virtual Reality, Motion Simulation and A 4D GIS

JUERGEN ROSSMANN, ARNO BUECKEN, MARTIN HOPPEN, MARC PRIGGEMEYER

Rossmann, J., Buecken, A., Hoppen, M., & Priggemeyer, M. (2016). Integrating Virtual Reality, Motion Simulation and a 4D GIS. *Research In Urbanism Series*, 4(1), 25–42. doi:10.7480/rius.4.854

Abstract

Geodesign requires the visualization of concepts and ideas within a context of geo-information of the respective place in a way that is understandable to people with different backgrounds – planners, geographers, architects, but also the users or inhabitants of the place. All of the roles involved have different requirements and need different information to fulfil their tasks within the geodesign process. In this contribution, we present the structure of a software system combining a GIS, a simulation system and a VR component, as well as interfaces to different interaction devices (like a GPS receiver, a spacemouse, multi-screen projection systems or devices for haptic feedback). This enables simulations of the place in its geographical context, as well as immersive presentations that are understandable regardless of the knowledge of a plan's symbolic language. All this happens without the need to convert frequently between the software tools that are commonly used by the different roles.

KEYWORDS

Virtual reality; Motion simulation; 4D GIS; Geodesign; Virtual Forrest

1. INTRODUCTION

According to Steinitz (2012) Geodesign "is the development and application of design-related processes intended to change geographical study areas in which they are applied and realized". He states that collaboration between the different roles involved in a geodesign process can be challenging. Therefore an intuitive visualisation of the steps and results is an essential part of a successful geodesign process. This goes together with the evolution of geographic information systems (GIS) from 2D to 3D, which happened in recent years, By now, some approaches even consider time as a fourth dimension. Visualization of the corresponding data moved ahead from simple maps towards three-dimensional landscapes and cities. However, in most cases, it is limited to a single display on a single computer.

In this contribution, we present an approach of a fully integrated 4D geographic information and virtual reality (VR) system. While data management is based on OGC (Open Geospatial Consortium) standards like the Geography Markup Language (GML), it supports the synchronization of multiple clients and allows rendering views for multiple screens – even for a seven screen panoramic projection system or a CAVE environment. Besides VR-style visualization techniques, it is also possible to use the data for simulation or to even feel it with a motion simulator system.

We will describe the synchronization of multiple computers, the visualization component and the use of a highly versatile motion simulator, which is based on an industrial robot. Several aspects like the used washout filter and the physiological foundations that enable the use of a robot with a still limited workspace to display poses and forces in a large world are introduced.

The rest of this paper is structured as follows: The next section illustrates, how we extend a 3D simulation system by GIS functionality. In the next section, we describe the simulation system's VR capabilities. Subsequently, the motion simulation approach is presented. The contribution ends with a conclusion of the presented work.

2. CREATING A GIS FROM A 3D SIMULATION SYSTEM

Most software environments require multiple independent components for editing and displaying 3D geo-data. While standard GIS provide a 2D top view on the scenery and in some cases a 2.5D or even a rather limited 3D view, additional software is needed to display the same scenery on large scale displays or in virtual reality systems. The disadvantage of multiple software products in the tool chain from data editing to the impressive visualization is the frequent need to convert and exchange data.

The presented approach was developed for a forest information and simulation system that required a number of algorithms and user interface approaches, which were already implemented in VEROSIM, an existing 3D simulation system. Originally, this system was developed for the simulation of robotic work cells, but has since been enhanced to a variety of applications from the fields of industrial automation, space robotics and environment. Thus, for the latter, instead of starting with a standard GIS, support for standard geo-data modelling and interfaces for geo-databases like SupportGIS Java (SGJ) or native PostgreSQL/PostGIS were added to the existing VEROSIM.

2.1 Flexible Database Interface

For accessing (geo-)databases from a simulation system, a flexible yet efficient concept was developed (Hoppen, Rossmann, Schluse, & Waspe, 2010). Its basic idea is to synchronize a simulation system's internal runtime database with the central database on schema, data and functional level. Using schema synchronization, the simulation system adopts the schema from the central database once during system start-up so both systems "speak the same language". Subsequently, data conforming to this schema can be replicated to the simulation database, on-demand. For example, driving a virtual car through a large forest model, spatially nearby data (e.g. surrounding trees, tile data) is loaded into the simulation database while objects are unloaded when they are left behind. Thus, the simulation database can be seen as an "intelligent", real-time capable cache for the central database. The approach also allows modifying replicated data. Here, change notifications are used to synchronize updates between the databases (Hoppen, Waspe, Rast, & Rossmann, 2014). Thus, when synchronizing multiple simulation clients to the same central database, it cannot only be used for data management, but also as a communication hub for the shared simulation model.

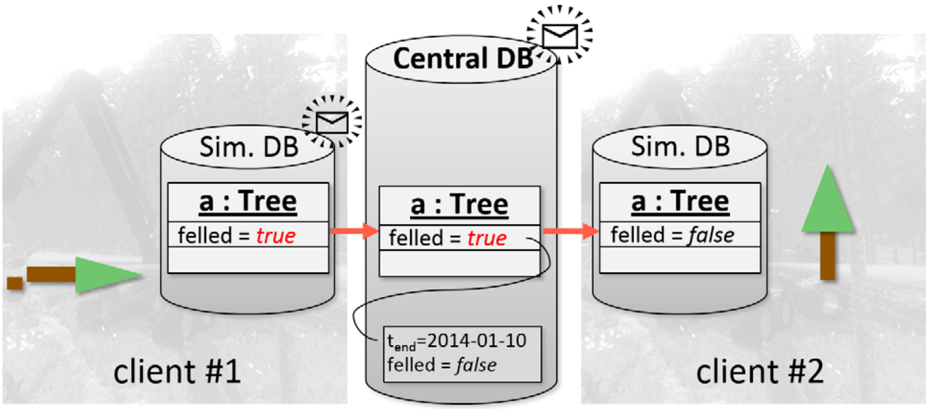

Figure 1. The architecture of the database synchronization approach.

In Figure 1, an exemplary synchronization scenario is shown. Two simulation clients with their respective databases (#1 and #2) are connected to a central database containing a simple ‘Tree’ object with a ‘felled’ state. The object is replicated to both clients. Client #1 changes the state to ‘true’ and syncs it to the central database where the previous value is versioned with a timestamp tend. Subsequently, a notification sent to client #2 allows it to adopt the change. Finally, besides schema and data synchronization, functional synchronization is used to translate semantics between the systems. For example, the transformation matrix of a CityGML (Kolbe, Gröger, & Plümer, 2005) implicit geometry is translated to a data structure known to the simulation system’s render engine. Altogether, for the presented work, this database synchronization concept was realized using the VEROSIM Active Simulation Database (VSD) and SGJ.

2.2 Temporal Data Management

Additionally, basic GIS functionality required for editing the stored data was implemented, e.g., for measurement, vector or raster editing, or gradient visualization, yielding a 3D simulation system with an integrated GIS. In order to capture and reproduce the changes of the 3D forest model (or any other model), we added time as a fourth dimension (Hoppen, Schluse, Rossmann, & Weitzig, 2012). This is realized by using a temporal database (Jensen, & Dyreson, 1998) as the central geodatabase. When changing data in a temporal database, its previous state does not get lost, but is preserved in terms of historic versions that are still accessible by the user. As geo-data represents the state of real world phenomena at one or more points in time, a temporal database allows capturing this inherent time dependence. Different interpretations of time, so-called time dimensions, may be applied. A transaction time database automatically associates committed timestamps with any change. In contrast, in a valid time database, the user (or some process) assigns timestamps that represent the point in real time a change has taken place or will take place. Both concepts can even be combined, yielding a bi-temporal database. Using the aforementioned database synchronization concept, a temporal snapshot is replicated to the simulation system’s runtime database. For that purpose, the user specifies a reference time within the simulation system. On changing this reference time, the snapshot gets updated accordingly. When altering the replicated data within the simulation system’s database, synchronized changes are versioned within the central, temporal database. Figure 1 shows an example, where a change from a simulation client is replicated to the central temporal database. Here, the previous value (“false”) becomes a historic version before the new value is adopted and a notification is emitted.

2.3 Client Synchronization

Note that all these mechanisms are independent of the actual data model and can be transferred to other (geo-)data than trees and forests. Thus, the very same approach can also be used to monitor the change in urban areas, e.g. using the CityGML data model, but also other common formats like IFC (Industry Foundation Classes), STL (STereoLithography) and many others are supported.

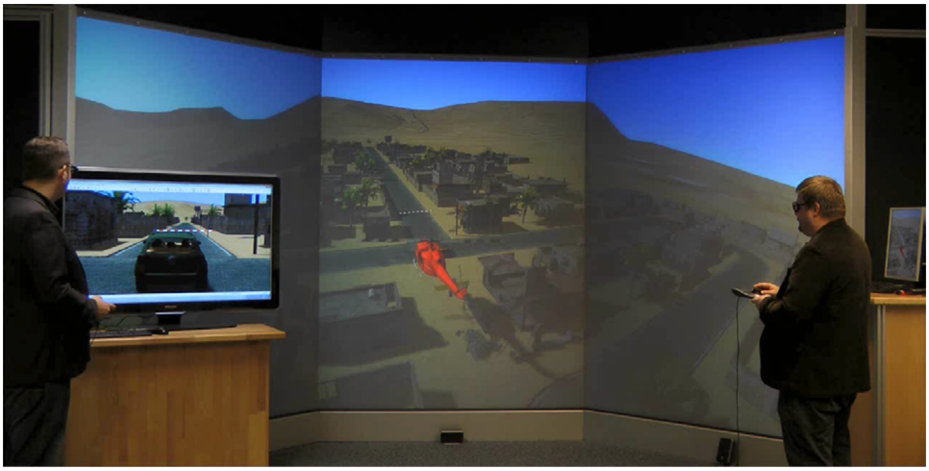

Figure 2. Example scenario for database synchronization.

Furthermore, our approach allows the synchronization of multiple independent clients either by a simple property synchronization mechanism, a fully-fledged distributed simulation protocol, or by using a central, active geo-database (Hoppen et al., 2014) as presented above. This allows to either link multiple displays for CAVE environments or multi-projector panoramic projections, or to generate independent views of the same scenery for several users. The approaches can even be combined so that each connected client can in turn use multiple screens. Figure 2 shows an example using combined synchronization approaches: the simulation model is managed by a central geo-database and replicated to two clients (jeep on TV, helicopter on multi-projector panoramic projection). The projection system itself distributes the simulation to six slave systems for rendering the three stereo images.

3. VIRTUAL REALITY

There are multiple virtual reality applications that deal with geo-information. Examples are landscape visualization, architecture as well as simulation of cars, aircrafts or other vehicles. In most cases, it is required to export the geo-data and convert it to some 3D format like DXF (AutoCAD Drawing Exchange Format) or IGES (Initial Graphics Exchange Specification) in order to use it in a simulation system. In the presented system, however, the integrated 3D renderer of VEROSIM is activated, the view is changed from 2D orthogonal to a 3D perspective projection and – if required – multiple computers are linked together with the synchronization approach described above. On each computer, the view frustum can individually be adjusted, allowing to define stereo views (where two computers render the images for the right and left eye), panoramic views (using adjacent screens) as well as a combination of both. The renderer supports different lighting situations, change of daytime, different weather scenarios and even the photorealistic visualization of natural objects. Figure 3 gives an example with a virtual city guide. The performance of the renderer scales with the hardware of the computer. On a standard PC with a graphic board designed for computer games it is possible to visualize environments with eighty million vertices at forty frames per second. 2D geography features can simply be projected on a 3D ground. It is also possible to use metaphors for this information – like an auto-generated fence for a surface feature representing property boundaries.

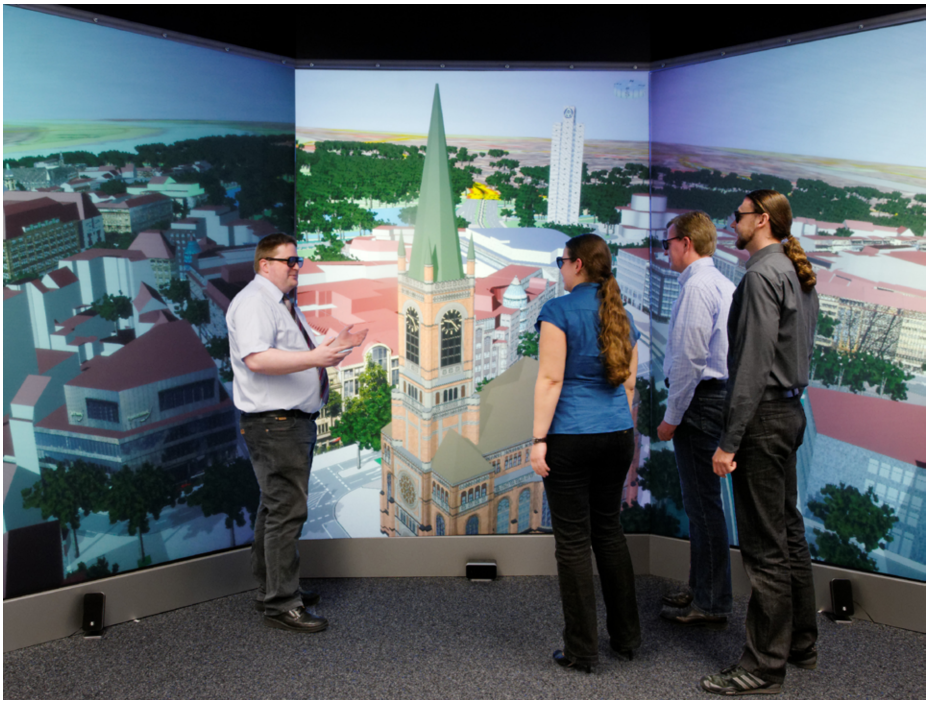
Figure 3. A virtual city guide.

The software system also offers physical simulation of objects. This way, it is possible to use the geo-data for simulation purposes. The objects can be controlled by the user with several different interaction devices like a wireless six DOF (degrees of freedom) tracking system, data gloves, a spacemouse, joysticks or even a dedicated hardware like a harvester seat with the manufacturer's on-board computer. The simulation can be configured in a way that the results or performance logs are also geo-coded. Therefore, it is possible to study and discuss results of the simulation in the corresponding environment. Examples of geo-coded logs are track-marks or geo-coded notices which can be displayed as a small sign.

4. MOTION PLATFORM

Although providing a deep visual immersion, it is still difficult, e.g., to estimate the real inclination of a hill or the effort needed to follow a steep road. This impression, however, can easily be achieved by moving the user according to the hill's slope. There are several approaches for this so-called motion cueing. Conventional approaches mostly utilize a Stewart/Gough platform, a hydraulic hexapod that allows moving the top plate in six dimensions. This system is scalable from a small installation, which can only carry a seat with a single person and a data helmet, towards a solution that moves a flight deck or a ship's bridge for professional multi-user training applications.

Due to their mechanics, all Stewart/Gough hexapods are limited in their rotatory movement. Thus, steep inclinations cannot be simulated with these devices. A more advanced approach is a motion simulator based on an industrial robot. While the main disadvantages against the hexapod are a lower maximum payload and a larger space requirement, the motion simulator benefits from the versatile movements of the industrial robot. With this device, even an overhead situation becomes possible.

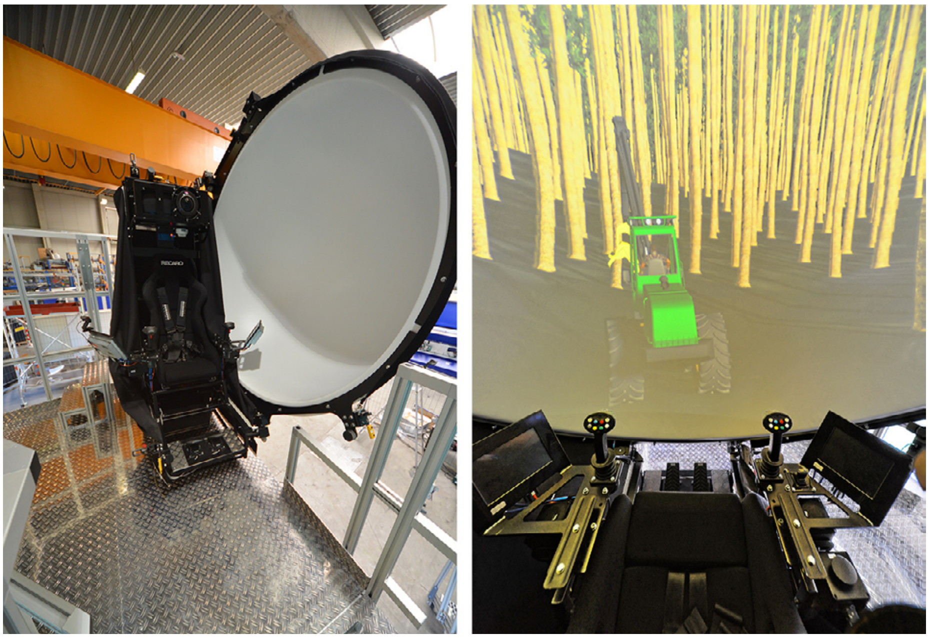

Figure 4. The capsule of the motion simulator.

For the work described in this contribution, we decided to use a robot-based motion simulator with a small capsule, which is equipped with a 180 cm wide hemispherical screen in front of a high resolution 3D projector with a fisheye lens above the user as well as a stereo sound system (Figure 4). For user interaction, the capsule provides two touchscreens, which can be used to display instruments or additional information, two three-axis joysticks, two pedals and a throttle control. The capsule is equipped with an opaque textile cover that keeps out visual impressions from the outside providing a better immersion.

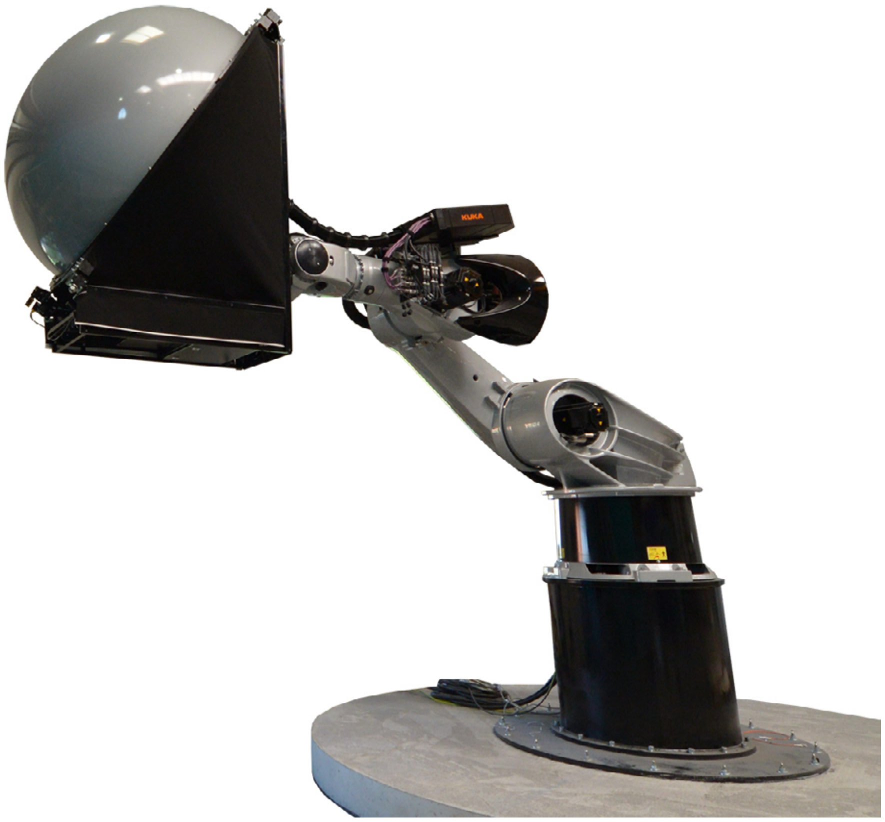

Figure 5. The motion simulator system.

Our system features a KUKA KR-500 TÜV robot, a six-axis industrial robot with a maximum payload of 500 kg. This is sufficient for the capsule including all installed electronics and a passenger with up to 120 kg (Figure 5). The robot operates in a work cell with a diameter of approximately 10 m and requires a height clearance of about 7 m. The capsule can be accessed in a height of 2.4 m by using a staircase and a retractable platform. To ensure the user's safety, the capsule is equipped with a bidirectional audio link, a video downlink and a smoke detector. The robot is hardware-restricted in its movements to avoid collisions between the capsule and the ground as well as between the capsule and the robot. Furthermore, the robot control includes an acceleration limit to control the forces that are applied to the user. The impacts are limited according to DIN EN 13814 (DIN, 2004), but still the user feels accelerations up to 2 g.

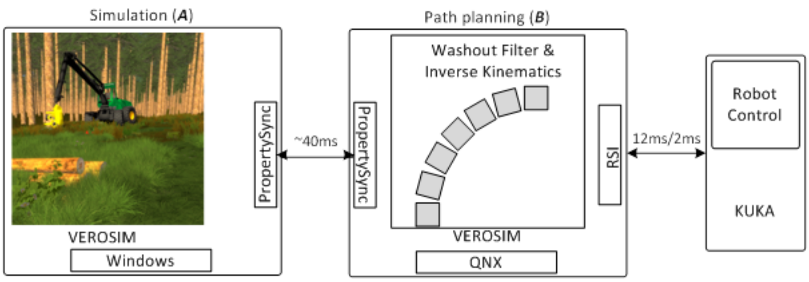

Figure 6. Components involved in motion cueing.

4.1 Motion Cues

The benefit of this six-degrees-of-freedom robot is the precise mapping of simulated movements and accelerations onto a pose in the real world. This way, an attached seat can be positioned and oriented according to the user's situation within the simulation. For example, in the simulation of a wood harvester (Figure 6, left), when a driver navigates through a forest, passing through rough terrain and evading obstacles, it is invaluable for a realistic simulation to extend the 3D visual information with a tactile element. In contrast to smaller screens, the hemispherical projection provides for the possibility of strong peripheral visual cues. These motion cues are caused subconsciously by motion visually observed in the peripheral field of view. Different researchers have already put huge efforts into studying these effects. Their conclusion is that these cues already provide a very strong motion feedback, but they can still be amplified by motion observed through the vestibular system (Telban, 2005).

The vestibular system responds differently to translational and rotational motion. Due to the robot's construction, there is a strong influence on the semi-circular canals and, therefore, motion perception due to extended rotations is intensified by design. Nevertheless, since thresholds for the human motion perception have been proposed (Zacharias, 1978), strong jerking movements can be utilized to convey translational accelerations by stimulating the otolithic organs.

By computing the appropriate robot motion in real-time, the driver's motion and the visual feedback can be synchronized for a holistic driving impression (Telban, 2005). This can be achieved by estimating motion cues caused by a vehicle and induced to a passenger. By calculating accelerations of the passenger frame with a subsequent washout filtering, a motion subset can be estimated that can be used to stimulate a passenger's vestibular system causing the relevant motion cues. Additionally, the passenger perceives the visual motion cues due to her peripheral field of view.

The washout-filtering step is essential to provide measures for the limited workspace of the robot (Grant, & Reid, 1997). Since a vehicle in the real world can move freely, with regards to its physical behaviour, while the passenger seat's motion envelope in the simulator is rather limited due to the constraints of the robot, a mapping of the motion has to be applied. Models of the human inner ear can be applied to preserve the perceived motion. Since the semi-circular canals as well as the otolithic organs can be modelled as damped systems (Zacharias, 1978), only parts of the actual motion are perceived until the "washout" masks others. Thus, it is possible, e.g., to stop accelerating the capsule in one direction after a short period of time and then to slowly move backwards to the pristine position for further motion induction without the passenger noticing.

This motion has to be performed by an actuator, e.g., a six-axis robot like our motion simulator, carrying a capsule providing a passenger seat. For this robot to move, an interface to the manufacturer's robot control system has to be provided. In general, such an interface needs to meet hard real-time constraints specified by the manufacturer, which are specific to a particular robot control system. Communication protocols, simulation, planning and execution are separated onto different machines constituting a distributed simulation. This way, time critical parts of the software are detached from the non-time-critical parts and executed on dedicated computers providing enough computing resources to meet the specified constraints.

4.2 Distributed Simulation

The simulation comprises different computers to carry out specific tasks (Figure 6). Computer (A) either runs a Windows or a Linux operating system. It provides a platform for VEROSIM to simulate a vehicle's dynamics and the environment it interacts with. For a realistic simulation, the vehicle's model is composed of different parts with different masses. A wheel suspension modelled as a damped mass-spring system is attached to provide realistic interaction between rough terrain and the vehicle (Jung, Rast, Kaigom, & Rossmann, 2011). In the example of the wood harvester, the vehicle also has a crane to grab and work on tree trunks. As this provides a huge level of interaction between vehicle and environment, it also provides a huge potential for a realistic motion feedback.

The motion feedback is calculated on computer (B) by transforming velocities and accelerations into new poses and robot trajectories. It runs the QNX operating system with its pre-emptive scheduler (Hildebrand, 1992). This is a mandatory component of the whole setup to meet the real-time constraints imposed by the Kuka Robot Control (KRC2). A sampling frequency of 1/12kHz has to be attained to achieve smooth robot motion. Thus, the QNX system clock runs with a 1 ms resolution to provide a fine granularity for the VEROSIM task scheduler.

To cope with the inverse kinematics and path planning, the simulation system VEROSIM uses a model of the physical robot and its environment. This model is used to evaluate the motion, considering the available workspace and robot constraints. These constraints are imposed by limits for the axis' positions, velocities and accelerations. By exceeding any of these limits, the robots movement is stopped and an error message is issued to signal a fault state. Because this also prevents the simulation from continuing, the robot constraints have to be strictly adhered to, to prevent a passenger from uncomfortable accelerations.

When a target pose is calculated that exceeds the robot limits, steps have to be taken to move the robot in a way to minimize the wrongly perceived motion cues. One examined approach was to move the robot to its farthest possible Cartesian position and then only continue moving those axes that are sufficiently far away from their limits to provide a smooth stop.

Two different approaches were implemented to compute velocity profiles for the robot to follow. When a velocity profile is calculated, it is used to interpolate new target positions for the robot's axes. These positions are set by the internal position controllers of the KRC2.

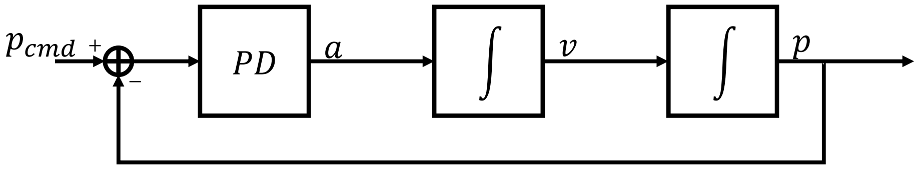

Figure 7. Linear control model for accelerated motion.

The first approach utilizes a simple linear model describing accelerated motion (Figure 7). Accelerations (a) are integrated twice (with the velocity (v) as intermediary) to provide positions (p). Since positions are controlled by accelerations, a closed-loop linear controller is necessary to ensure a stable system behaviour. The controller is implemented with the PD (proportional-derivative) control algorithm comparing the desired position (p_cmd) with the actual position (p). The controller is parametrized to provide small system response times while being stable with little overshoot. Subsequently, the acceleration values are truncated to fit the robot acceleration limits.

The second approach parameterizes cubic hermite splines to describe a velocity profile. When a new target position is received, it is taken as the spline's final point setting the velocity to zero. Accordingly, the robot's current position and velocity is used for the spline's starting point. This allows coping with two scenarios. First, no new target position is received until the spline's final point is reached. Thus, the robot is stopped and waits for new commands. Second, a new target position is received while the robot is still moving. Since the current position and velocity is used to calculate a new profile, the current one can be replaced by the newly computed velocity profile without causing jerking behaviour.

Either way, resulting poses of the simulated robot are sampled with a frequency of 250Hz and subsequently used as target values for the internal position controllers of the KRC that move the physical robot accordingly.

To provide target positions for the previously described approaches, a washout filter was implemented being the basis for the whole motion cuing process. To this point, a classical washout filter design (Krämer, 2004) has been utilized on computer (B), executed with a frequency of 250 Hz. A basic layout of this filter is depicted in Figure 8. Three main parts can be identified that operate in close coordination. For all of them, the vehicle simulation has to provide accelerations (a,) that are separated into a translational and a rotational component. The translational component is utilized by the tilt coordination (TILT, Figure 8) as well as the high pass filter (HP, Figure 8 top) for the translational motion. Since the translational high pass filter removes low frequency components, which would be useful for long acceleration phases in vehicles, the tilt coordination transforms these low frequency accelerations into an orientation that is applied to the passenger seat. This way, the earth's gravitational force is used to display accelerations that would otherwise be lost.

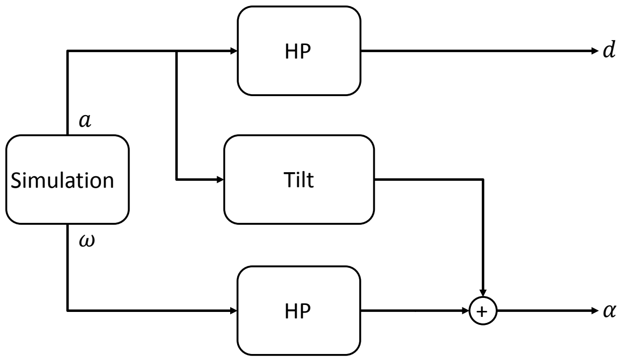

Figure 8. Basic washout filter principle.

These changes in orientation will ultimately induce rotational motion cues if performed too rapidly. Since thresholds for the perception of these changes exist, the effects can be compensated. As with translational accelerations, rotational accelerations are also high-pass-filtered (HP, Figure 8 bottom) to remove unacceptably long acceleration phases that would cause the robot to move outside its predefined limits. However, low frequency components are processed differently as there is no counterpart to the tilt coordination. Thus, they are lost and cannot be utilized in motion cueing.

The filtered accelerations (d,) can be used to control the robot motion. To guarantee the capsule to return to its pristine position, a linear controller is used. The controller is parameterized to keep accelerations for this motion below perceptual thresholds to avoid the induction of false motion cues. The pristine position has to be chosen carefully to provide for a maximum flexibility in the consecutive movements. The aforementioned lengthy acceleration phases might exhaust the robot's workspace reducing otherwise possible motion cues.

4.3 Synchronization

Different computer systems are combined to provide a distributed simulation for our motion simulator. All computers have a specific task to perform and, therefore, have to synchronize specific properties. As mentioned above, VEROSIM provides a simple property synchronization mechanism (Hoppen et al., 2014). It can be used to replicate individual properties of the simulation model between VEROSIM instances on different computers. Every time a property is modified, the new value and its timestamp are sent to all other connected instances.

The VEROSIM instance on computer (A) simulates a vehicle whose seat's frame is used as an input frame for the washout filter. As the washout filter is executed on the QNX VEROSIM instance on computer (B), the seat's frame has to be synchronized between both instances. Hence, computer (A) acts as a server for computer (B) and changes of the seat's frame are sent to computer (B) where the appropriate accelerations for the translational and rotational components are calculated for further processing.

As this synchronization is a non-time critical part of the simulation, frames are not sampled at fixed rates, but rather as soon as changes occur. Therefore, as the vehicle simulation on computer (A) is carried out with an update frequency of 25 Hz to 30 Hz, new samples arrive at the least every 40 ms.

4.4 Robot Control

For its KRC2 robots, KUKA provides a control panel that can be used for robot control. It is a graphical user interface to display variables, states and system information. It can also be used for writing KRL (Kuka Robot Language) programs to be executed on the robot's real-time hardware. Furthermore, it allows to directly moving the robot with a simple keystroke. Even though this enables the user to easily manipulate the robot's position, it is not sufficient to implement large sensor applications or applications dependent on external path planning.

For such applications, KUKA provides the Robot Sensor Interface (RSI) technology package. It is a KRL application-programming interface comprising function blocks that can be connected into block diagrams implementing complex algorithms. On the KRC2, the RSI programming is carried out using KRL code to instantiate blocks and manually connect them by function calls. When the RSI definition is finished, a function call can pass control over the robot position from the integrated control panel to an external system by entering the so-called sensor driven mode. This mode enables the RSI function blocks and, therefore, causes an override of the user interface.

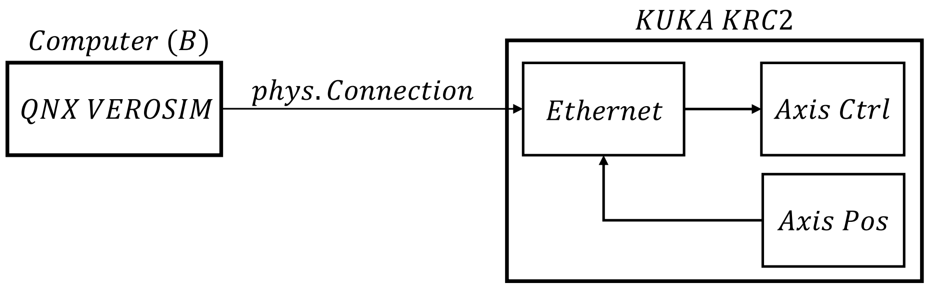

Figure 9. Position control integration with the Robot Sensor Interface.

For our motion simulator, a block diagram as depicted in Figure 9 (right) is implemented. A function block is instantiated for Ethernet connectivity (Ethernet), axis control (Axis Ctrl) and position sensor (Axis Pos). This way, new positions can be commanded via Ethernet to be applied to the robot's axes. The axis control block passes positions to the internal position controller and makes the robot move. The robot's current position is also transmitted back to the external system (QNX VEROSIM) via Ethernet (Figure 9 left). Thus, here, a closed-loop feedback controller for the motion feedback algorithms can be implemented on Computer (B) as well.

On the QNX computer, VEROSIM is executed as a real-time process and is connected to the KUKA KRC2. As mentioned above, its primary task is to execute a simulation of the robot kinematics and to calculate velocity profiles for the physical robot to follow. Thus, the QNX VEROSIM opens a data channel to the Ethernet function block running in the RSI block diagram and replicates the simulated robot's motion by commanding positions sampled from the velocity profile. Since the RSI Ethernet block demands a 12 ms time slot, new positions are sampled asynchronously with a 1/12 kHz frequency.

A combination of these techniques yields a simulation system that provides holistic driving impressions. A passenger driving a simulated jeep (Figure 10) can see and feel accelerations by steering a vehicle through simulated terrain in a realistic manner. For example, while driving a long left-hand curve the motion simulator capsule is moved likewise to induce the realistic driving impression. Figure 11 gives an impression of the motion simulator's dynamic behaviour.

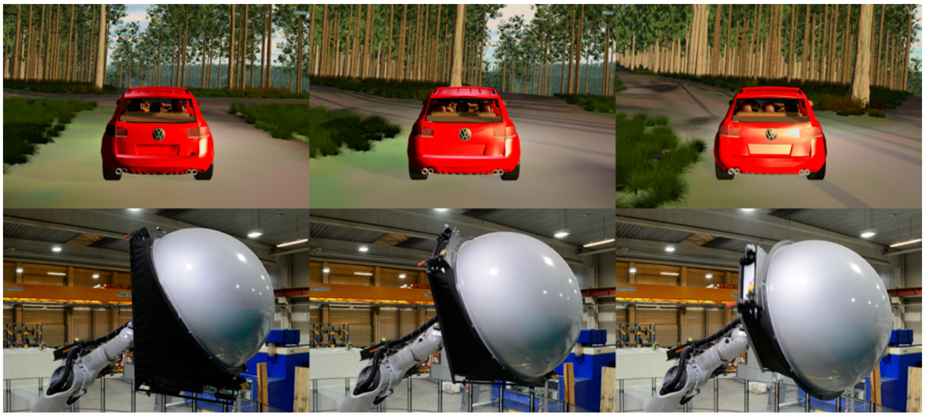

Figure 10. Motion simulator responds to left-hand curve of simulated jeep.

Figure 11. An impression of the motion simulator's dynamic behaviour.

5. CONCLUSIONS

Altogether, the presented combination of a 3D simulation system with VR, GIS and motion simulation functionalities provides a fully integrated virtual reality and GIS approach. It reduces the usually required tool chain to a single software, eliminates the need to permanently import and export data between multiple systems and adds the possibility to interact with the geo-data like in a common GIS even when displayed in a VR system. It becomes possible to change object data in the GIS, even to generate live maps from the geo-data and to immediately explore the results in the virtual reality environment. On the other hand, georeferenced simulation results can be evaluated with all available GIS tools, which is one of the ideas of a Virtual Testbed (Rossmann, Jung, & Rast, 2010). For example, performance logs of a harvester can be combined with 2D or 3D maps and displayed in the GIS or VR view.

The haptic feedback from an industrial-robot-based motion simulator adds even more information about the geo-data to the impressions of the user. Movements of the camera or of any simulated object within the geo-data are converted to robot movements by using washout filters which consider the limits of the physiological movement perception of a human. These calculated movements are then passed to the robots inner control loop in a real-time process. Together with the visual impacts of the visual-range-filling hemispherical projection, this haptic feedback provides a holistic impression for the user.

The motion feedback induced to the passenger's perceptual organs is directly depending on the path and motion planning carried out on the real-time system. A huge variety of different parameters and perceptual limits can be exploited to increase the immersion of a passenger into the simulated environment, holding the capabilities for future research. A washout filter implementation specifically applied to the robot axes in favour of the Cartesian filter implementation will lead to a more efficient workspace utilization. Hence, specific feedback channels can be prioritized (e.g. lateral motion) leading to an enhanced motion perception, while simultaneously reducing the risk of motion sickness.

With the presented approach, GIS, VR visualization, simulation and haptic feedback are merged together, delivering added value for each of these fields. In geodesign processes, this combination provides a foundation for information exchange between the different roles. The VR system allows an intuitive and immersive visualisation of the planned concepts and therefore provides access to the displayed information for every role without the necessity to learn and understand special drawings or the symbolic language of other roles.

Applications for this integrated solution with a robot-based motion simulator or a Stewart/Gough platform range from city visualizations including virtual city tours and presentations including cars or helicopters to simulations like the presented wood harvester and Virtual Testbeds in multiple areas of engineering. With the 4D support, change visualisations of larger landscapes become possible.

ACKNOWLEDGEMENTS

Research project Virtual Forest: This project is co-financed by the European Union and the federal state of North Rhine-Westphalia, European Regional Development Fund (ERDF). Europe – Investing in our future.

REFERENCES

DIN – German Institute for Standardization (2004). DIN EN 13814: Fairground and amusement park machinery and structures – Safety; German version EN 13814:2004. Beuth-Verlag.

Grant, P. R., & Reid, L. D. (1997). Motion washout filter tuning: Rules and requirements. Journal of Aircraft, 34(2), 145-151.

Hildebrand, D. (1992, April). An Architectural Overview of QNX. In USENIX Workshop on Microkernels and Other Kernel Architectures (pp. 113-126).

Hoppen, M., Rossmann, J., Schluse, M., & Waspe, R. (2010). A New Method For Interfacing 3D Simulation Systems And Object-Oriented Geo Data Sources. In T. Kolbe, G. König, & C. Nagel (Eds.), The 5th International Conference on 3D Geoinformation ISPRS 2010 (pp. 51-56). Aachen, Germany: Shaker Verlag.

Hoppen, M., Schluse, M., Rossmann, J., & Weitzig, B. (2012). Database-Driven Distributed 3D Simulation. Proceedings of the 2012 Winter Simulation Conference (pp. 1-12).

Hoppen, M., Waspe, R., Rast, M., & Rossmann, J. (2014). Distributed Information Processing and Rendering for 3D Simulation Applications. International Journal of Computer Theory and Engineering (IJCTE), 6(3), 247-253.

Jensen, C. S., Dyreson, C. E., Böhlen, M., Clifford, J., Elmasri, R., Gadia, S. K., et al. (1998). The consensus glossary of temporal database concepts—february 1998 version. In Temporal Databases: Research and Practice (pp. 367-405). Berlin-Heidelberg, Germany: Springer.

Jung, T. J., Rast, M., Kaigom, E. G., & Rossmann, J. (2011, January). Fast VR Application Development Based on Versatile Rigid Multi-Body Dynamics Simulation. In ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 1481-1490). American Society of Mechanical Engineers.

Kolbe, T. H., Gröger, G., & Plümer, L. (2005). CityGML: Interoperable access to 3D city models. In Geo-information for disaster management (pp. 883-899). Berlin-Heidelberg: Springer.

Krämer, M. A. (2004). Universelle Fahrzeugsteuerung als integrativer Bestandteil einer VR-Simulationsplattform. Shaker Publishing.

Rossmann, J., Jung, T. J., & Rast, M. (2010, October). Developing virtual testbeds for mobile robotic applications in the woods and on the moon. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on (pp. 4952-4957). IEEE.

Steinitz, C. (2012). A Framework for Geodesign – Changing Geography by Design. Redlands, CA: Esri Press.

Telban Robert, J., & Cardullo, F. M. (2005). Motion Cueing Algorithm Development: Human-Centered Linear and Nonlinear Approaches. NASA/CR-2005-213747.

Zacharias, G. L. (1978). Motion cue models for pilot-vehicle analysis. Bolt Beranek and Newman inc. Cambridge MA Control Systems Dept.