

An architecture for driving simulator databases with generic and dynamically changing road networks

Armin Kaussner, Martin Grein, Hans-Peter Krüger, Hartmut Noltemeier

IZVW (Center for Traffic Sciences at the University of Würzburg)
Roentgenring 11
97070 Würzburg, Germany
e-mail: kaussner@psychologie.uni-wuerzburg.de

Abstract

For the various demands of the experiments carried out in the IZVW driving simulator a new concept for databases was developed. The road network and the surrounding features of landscape and terrain, flora and buildings are represented in a topological way. Through this the geometrical representation is computed during the simulation in a small area surrounding the driver, including all that is visible for the driver. This generic method offers various possibilities for scenario design. First of all, the extent of the road network is almost unlimited because of the efficient representation. Further on, it can be changed during the simulation when outside the view of the driver. This modification is either carried out interactively by the researcher or automatically by the system itself in supervising conditions freely to be defined for the variables (e.g. physiological measures) available in the simulation.

Introduction

Usually the design of scenarios for experiments in driving simulators is carried out in the following steps (see several contributions in [4], [5]):

Firstly, the road network and its surrounding landscape is defined. On a purely graphical level all elements have to be specified. This geometric information is used by the image generators to compute the image presented to the driver. Furthermore, some modules of the simulation need information about the lanes and their geometry, type of paving, traffic rules etc. This data is stated in the logical road network. The total of the geometrical and logical information will be referred to as “database” throughout the paper. A scenario is given if the researcher plans certain events (e.g. by orchestrating other vehicles) that take place in such a database.

Traffic research at the IZVW made it necessary to formulate new requirements for scenarios, e.g.:

- On examining workload, the driver has to accomplish a specific sequence of situations. However, at traffic junctions such as crossways the driver might choose a turn-off that does not lead him to the next situation of the experiment. To prevent the interruption of the experiment the database should be able to automatically change the road network in a way that nonetheless the next situation planned by the experimenter will follow.
- Another requirement could be that the sequence of situations is controlled by external criteria like physiological measures or by different states of the driver. It should also be possible to determine the sequence of situations by an online step to step decision of the experimenter.

All these examples require that the road network can be modified during simulation. In the most extreme case the road network is generated throughout the course of the experiment. Obviously these requirements cannot be fulfilled by predefined fixed databases. The concept of a generic and dynamically changing road network is described in the following.

Database architecture

Conventional databases have the property of global geometrical consistency. This means that their road network is determined non-ambiguously and can be drawn as a whole. The idea underlying our concept for databases is only to claim local geometrical consistency. Geometrical information is only computed in a small sphere around the driver. The size of this sphere is determined by his range of sight. As the road network and the landscape inside this sphere does not change, everything lying outside can be altered without being noticed by the driver. The database as a whole is represented in a topological way. In the following this representation and the algorithm that computes a local geometrical consistent road network is described. Details concerning the modeling of traffic rules and road roughness, for example, have been omitted.

Representation of the road network

The road network is a graph $RN = (V, E)$. The set of nodes V is partitioned into two parts: $V = COURSES \cup AREAS$.

A $v \in COURSES$ is a continuous road of arbitrary length without turn-offs. Also, the profile of the cross section (number of lanes, their width, type and driving direction) doesn't change inside of v .

A $v \in AREAS$ is a rectangular area containing lanes that model conjunctions between $COURSES$ with different cross section profiles, crossways, drives to highways etc.

Each $v \in V$ has ports $v.p_i$ that connect to other nodes of V .

For a $v \in COURSES$ there are two ports, labeled as $v.BEGIN$ and $v.END$. A $v \in AREAS$ has four ports, $v.TOP$, $v.BOTTOM$, $v.LEFT$ and $v.RIGHT$. An edge out of E always connects two ports of nodes out of V : $E = \{(v.p_i, w.p_j) : v, w \in V\}$

Figure 1 shows a part of a road network that is defined in this manner.

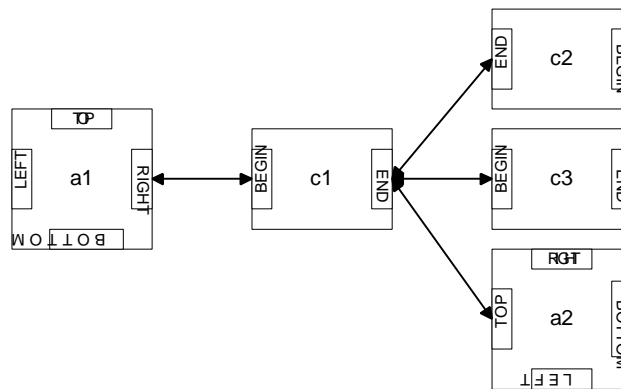


Figure 1: Topological representation of a road network

Remarks on the figure:

- $c_1, c_2, c_3 \in COURSES$ and $a_1, a_2 \in AREAS$.
- Plotting $a_2.TOP$ at the left border of a_2 has no geometrical meaning. The labels $TOP, BOTTOM, LEFT, RIGHT$ for ports of $AREAS$ simply refer to the position of the underlying rectangle during in the design phase of the area.
- The same holds for $COURSES$: They are designed from $BEGIN$ to END , but they can be connected opposite to this direction. An example for this is the connection of $c_1.END$ with $c_2.END$.
- Seen from $c_1.END$ the connections to $c_2.END$, $c_3.BEGIN$ and $a_2.TOP$ represent alternatives in the road network. They are selected at runtime, as described in the section “update propagation”.

The nodes $v \in V$ contain two pieces of information that are treated in the section „update propagation“ more closely. For sake of completeness of the definitions these terms are introduced here.

- During simulation the selection of a $w.p$ connecting to $v.p_i$ out of several alternatives is done by selection functions that are defined in the ports. The selection functions

save their result in $v.ActiveEdge \in E \cup \{NULL\}$. By default $v.ActiveEdge = NULL$, which means that no connection is active.

- Since position and orientation of a $v \in V$ is known only during the simulation due to the changeable road network, a transformation matrix $v.T \in \mathbb{R}^{4 \times 4}$ is stored in v . The matrix rotates and translates v . By default $v.T = \text{diag}(1,1,1,1)$.

Lanes

Each $v \in V$ has in addition to the ports a set of so called Lane cells, $v.LANECELLS = \{lc_1^{(v)}, \dots, lc_n^{(v)}\}$, that define the actual run of the lanes contained in v . $lc_i^{(v)}$ denotes the i th lane cell of v . A lane cell contains a set of lanes, $lc_i^{(v)}.LANES = \{l_{i,1}^{(v)}, \dots, l_{i,k}^{(v)}\}$. Thereby $l_{i,j}^{(v)}$ denotes the j th lane of the i th lane cell of v . Each of these lanes is defined by a parametric curve that can be seen as the ideal track of the lane. With it a lane $l_{i,j}^{(v)}$ of length S in every point $s \in [0, S]$ provides the following information:

- $l_{i,j}^{(v)}.p(s) \in \mathbb{R}^3$: Coordinates in the world frame
- $l_{i,j}^{(v)}.a(s) \in \mathbb{R}^3$: Tangent angles
- $l_{i,j}^{(v)}.k(s) \in \mathbb{R}^3$: Lane curvature
- $l_{i,j}^{(v)}.DIRECTION \in \{NORMAL, REVERSE\}$: Direction with regard to the curve parameter running from 0 to S .

The coordinates $l_{i,j}^{(v)}.p(s)$ are computed by transforming the points extracted from the parametric curve by the matrix $v.T$. The tangent angles $l_{i,j}^{(v)}.a(s)$ are adjusted by the angles of rotation that are contained in $v.T$.

In addition to this information about lane width, lane type, road roughness, traffic rules etc. is stored with the lanes.

Depending on the type of v its lane cells and the lanes in it can be specified more precisely.

In $v \in COURSES$ the set $v.LANECELLS = \{lc_1^{(v)}, \dots, lc_n^{(v)}\}$ is ordered. $lc_1^{(v)}$ is associated with $v.BEGIN$ and $lc_n^{(v)}$ is associated with $v.END$. The parametric curves that define the lanes $l_{i,j}^{(v)}$ of the lane cells are parallel to each other. Thus they can be derived from a single parametric curve, the so called reference curve, and the distances to it. The distances are given by the cross section profile of v . The reference curve can be a straight line, the sequence clothoid – circular arc – clothoid („bend“) or an arbitrary spline curve. The lanes of lane cells in $COURSES$ must fulfill the condition

$$l_{i,j}^{(c)}.a(S_{i,j}) = l_{i+1,j}^{(c)}.a(0) \quad \text{and} \quad l_{i,j}^{(c)}.p(S_{i,j}) = l_{i+1,j}^{(c)}.p(0)$$

$S_{i,j}$ denotes the length of the lane j in lane cell i . This means that the lanes of a lane cell must connect smoothly to the lanes of a neighbouring lane cell.

The left part of Figure 2 shows a $COURSE c$ consisting of three lane cells. The reference lane (drawn dashed) of $lc_1^{(c)}$ is a straight line, the one of $lc_2^{(c)}$ is a bent and the one of $lc_3^{(c)}$ again is a straight line.

In the case $v \in AREAS$ the set $v.LANECELLS$ can be seen geometrically as tiling of the rectangle underlying v . As shown in the right part of Figure 2 neighbouring lane cells have to be compatible with regard to their lanes. This means that lanes have to connect smoothly to the corresponding lanes in the neighbouring lane cells.

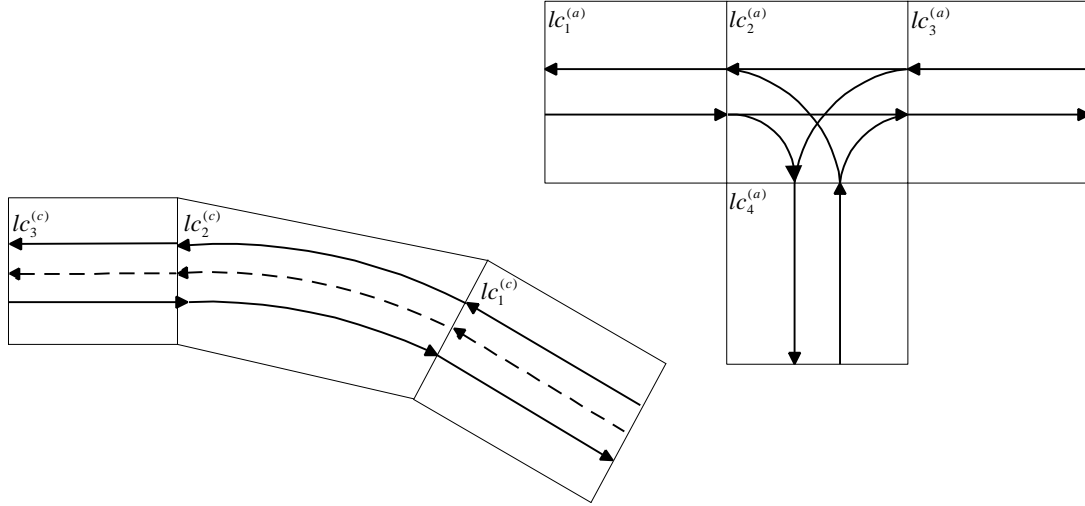


Figure 2: Lane cells of a COURSE (left) and of a junction modelled as AREA (right)

Update propagation

Due to the ambiguities in RN concerning the connections of ports the road network cannot be drawn as a whole by the image generators. The procedure described below is used to extract at each time point t_i of the simulation a graph $RN^*(t_i) \subseteq RN$ out of RN that is geometrical consistent inside a sphere around the driver.

If the driver has the position $p(t_i)$ at a time point t_i then this sphere is defined as

$$U_r(t_i) = \{u \in \mathbb{R}^3 : \|p(t_i) - u\| \leq r\}$$

The algorithm for computing $RN^*(t_i)$ is as follows:

Start of the simulation at t_0 :

The driver is put on a certain lane $l_{i,j}^{(vStart)}$ at position s ($0 \leq s \leq S_{i,j}$) by the researcher. For this, $v.T$ is computed such that

$$l_{i,j}^{(vStart)}.p(s) = (0,0,0) \text{ and } l_{i,j}^{(vStart)}.a(s) = (0,0,0)$$

This assumes that at the start of the simulation the vehicle resides in the origin of the world coordinate system ($p(t_0) = (0,0,0)$) and has a yaw angle of 0° .

Furthermore the initializations $l_i^{(vStart)}.t_{UPDATE} = 0$ and $lc.t_{UPDATE} = -\infty$ are made. Thereby $lc.t_{UPDATE}$ is an update timestamp that is stored in every lane cell. Its usage will be apparent in the procedure `update(.)` specified below.

The active edges of each port are set to $v.p_i.ActiveEdge = NULL$ which signals that the port has no active edges yet.

At each time step $t_i, i \geq 0$:

Firstly the lane cell clc (current lane cell) to which $p(t_i)$ has the smallest distance is determined. Starting from clc the graph $RN^*(t_i)$ is updated recursively through the call of `clc.Update(NULL)`. Below this procedure is described in a C++ like pseudo code. Optimizations that obviously are possible at several points of the procedure are omitted due to clarity.

```
Lanecell::Update(Lanecell lc_caller)
{
    // abortion of recursion if lanecell already was treated in this update step
    if (tUpdate == ti)
        return;

    // has lanecell not been treated in the last update step?
    if (tUpdate < ti - 1)
    {
        // compute transformation matrix T such that lanecell connects
        // smoothly to lc_caller
        lc.ComputeT(lc_caller);

        // maintain ur(ti)
        bool outside_ur = (lanecell lies completely outside of ur(ti));

        // if lanecell connects to one or more ports then loop over these
        // ports
        Port p = GetFirstAssociatedPort();
        while (p != NULL)
        {
            if (outside_ur)
            {
                // reset active edge of port for later updates
                p.ActiveEdge = NULL;
            }
            else
            {
                // select an edge out of the alternatives of port p
                p.SelectActiveEdge();
            }
            p = GetNextAssociatedPort();
        }

        // abort recursion, if lanecell lies completely outside of ur(ti)
        if (outside_ur)
            return;
    }

    // set update timestamp
    tUpdate = ti;

    // loop over lancells lc_neighbour that are neighbouring this lanecell (look
    // for them even beyond ports)
    lc_neighbour = GetFirstNeighbouringLanecell();
    while (lc_neighbour != NULL)
    {
        // recursion: update neighbouring lanecell
        if (lc_neighbour != lc_caller)
            lc_neighbour.Update(this);

        lc_neighbour = GetNextNeighbouringLanecell();
    }
}
```

Via the method `Port::SelectActiveEdge()` called from `LaneCell::Update(.)` the road network is made non-ambiguous by selecting the active Edge $v.p_i.ActiveEdge$ of a port $v.p_i$. This can be done in different ways:

- There is only one edge emanating from $v.p_i$. Then the active edge is set to this edge.
- There are multiple edges emanating from $v.p_i$ and the method `Port::SelectActiveEdge()` has access to variables and measures of the driving simulation. This is the case in the software framework outlined in [1], for example. Then the selection is carried out by a function $v.p_i.sel$ which is defined in ports and which selects the active edge based on conditions on measures taken during simulation. A simple example of such a function is:

$$v.p_i.sel(speed) = \begin{cases} (v.p_i, a_1.p_j) & \text{if } \left\{ \begin{array}{l} speed < 100 \text{ km/h} \\ else \end{array} \right. \\ (v.p_i, a_2.p_k) \end{cases}$$

Since inputs of the researcher during simulation are coded as variables that can be accessed by the method `Port::SelectActiveEdge()` the road network can be modified by the researcher interactively. Another example for this kind of selection is given in the section „applications“.

- There are multiple edges emanating from $v.p_i$ and there is no port selector defined. Then the first edge (in the order of definition) is selected.

As well, the method `Port::SelectActiveEdge()` makes sure that if $v.p_i.ActiveEdge = (v.p_i, w.p_j)$ was chosen then $w.p_j.ActiveEdge = (w.p_j, v.p_i)$ is set.

Scenario definition

With the proposed architecture for databases the researcher has the ability to design scenarios including the underlying database fitted for his experiment. Thus, he is able to describe the required items in a simple scripting language. This is the same scripting language that is used for configuring the software framework presented in [1]. Within this framework, the researcher can, along with the scenario, also prepare the remaining software modules of the simulator relevant for his experiment (sound, driver support systems etc.) in a consistent manner.

The design of a scenario essentially takes the following steps:

1. Definition of the nodes $v \in V$:

At first the lane cells of the nodes and their lanes must be defined. Therefore the researcher either chooses ready made nodes (e.g. crossways, drives to highways etc.) out of a library or he describes the lanes by himself. Compliance with several guidelines for building roads is monitored automatically. If essential for the experiment he can offend against these guidelines and, for example, build bents without clothoids.

2. Definition of lane information:

Unless this has already been done for the nodes that are chosen from the library the researcher provides the nodes with information about cross section profiles, traffic rules, traffic signs, road roughness etc.

3. Creation of surrounding landscape:

Unless the landscape is pre-defined for the nodes that are chosen from the library the landscape surrounding the nodes has to be defined. This can be done either by manually placing graphical objects (trees, buildings, hills, ...) or by assigning a landscape type to the

nodes. The landscape is generated randomly out of these types. The generation can be controlled by some characteristic parameters.

4. Connection of the nodes:

The researcher connects the nodes via their ports. If he plans alternatives in the road network he has to specify the port selectors. This is done by forming conditions for the variables and measures provided by the driving simulator.

5. Definition of the scenarios:

For this purpose the behaviour of the ambient traffic in several parts of the database is determined and the state or the parameters of software modules (e.g. the maximum possible deceleration of an ACC System) is changed depending on the position of the driver in the database.

Applications

This section presents two scenarios that take advantage of the proposed concept for databases.

Turning at a junction

During an experiment the driver faces the situation shown in the left part of Figure 3. At the intersection, he is given the instruction to find a sufficient gap in the east-west traffic. After which he must turn to the left. The joining road leads him to the next situation.

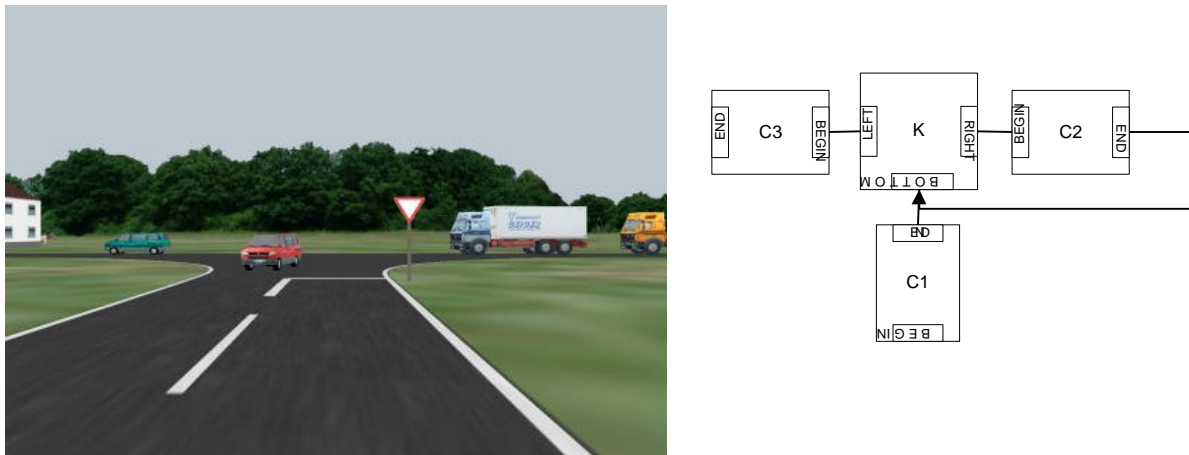


Figure 3: Crossway situation as seen by the driver (left) and its topological representation (right)

Of course it can happen (for example if the driver's attention is affected by alcohol) that the driver ignores the instruction of the researcher and takes the road leading to the right. If all the possible lanes were to be taken into consideration the planning of an experiment would be a complicated task. Such difficulties can be solved easily using the topological view on the road network. For the above mentioned situation, the researcher has to build the road network as shown in the right part of Figure 3. The driver reaches the intersection K via $C1$. If he correctly turns off, $C3$ will lead him to the next situation. If he turns off to the right he follows $C2$ until time point t_i where K leaves his range of sight $U_r(t_i)$. In this case the intersection K is automatically moved to $C2.END$ and the driver reaches the intersection again without noticing it. This procedure is repeated until he makes the correct turning.

Changing the road network depending on physiological measures

At the IZVW algorithms have been developed to detect the drivers state out of his eye closure level (see [2]). These states like „awake“, „tired“ and „sleepy“ can be accessed by the port selection functions described in section “update propagation”.

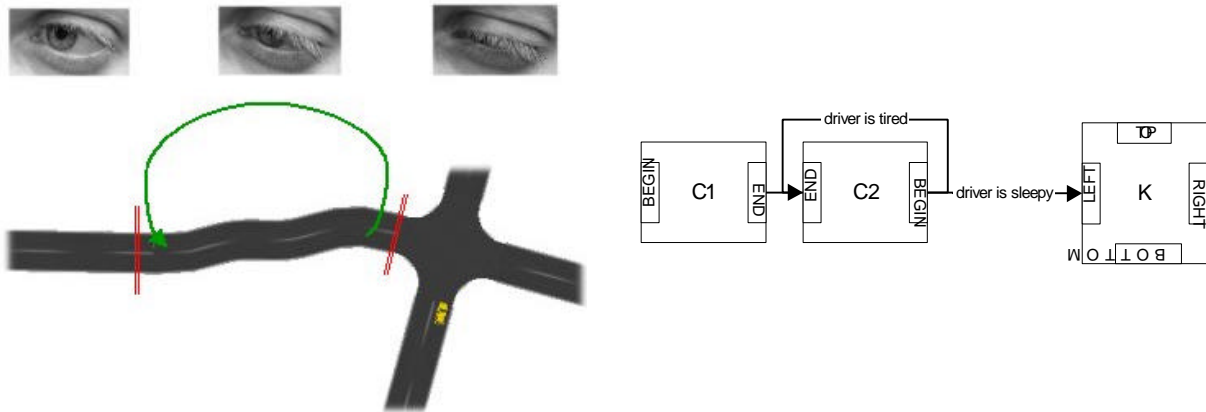


Figure 4: As soon as the driver is sleepy he encounters a junction.

Thus a scenario can be constructed as outlined in Figure 4. As long as the driver is awake a very monotonous route *C2* is chosen. As soon as the algorithm detects the state „sleepy“, the driver is confronted with an intersection *K*. At this intersection the yellow car will violate his right of way.

Conclusion

An architecture for driving simulator databases is presented that allows to modify the road network during simulation. Thereby a dynamic database is created which can easily be used by the experimenter and allows him to design a road network perfectly tailored for his problem. To control the degrees of freedom the driver has when choosing his way through the dynamic database, the researcher has to plan the road network from a topological point of view. The geometrical road network then is computed unnoticeable for the driver during simulation.

References

- [1] M. Grein, A. Kaussner, H.-P. Krüger, H. Noltemeier: A flexible application framework for distributed real time systems with applications in PC based driving simulators, Proc. DSC 2001, Sophia-Antipolis, 2001
- [2] V. Hargutt: Eyelid Movements and their Predictive Value for Fatigue Stages, Paper presented at the 3rd International Conference of Psychophysiology in Ergonomics, San Diego, 2000
- [3] H. Noltemeier: Graphentheorie: mit Algorithmen und Anwendungen, de Gruyter Lehrbuch, 1975
- [4] Proceedings of the Driving Simulation Conference, DSC1999, Paris, 1999
- [5] Proceedings of the Driving Simulation Conference, DSC2000, Paris, 2000