Road Vehicle Navigation through Virtual World Simulation

N. H. C. Yung, C. Ye and F. P. Fong
Department of Electrical & Electronic Eng.
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: nyung@hkueee.hku.hk

Keywords: virtual world, real-time rendering, navigation

ABSTRACT

In this paper, an integrated virtual world simulator for road vehicles and networks is presented. The structure of the simulator is modular and object-oriented, where the virtual world is hierarchically constructed. It supports 2D/3D real-time graphic rendering of objects which can be visualized on multiple X-windows, and a direct 'plug-and-play' of algorithms written in C/C++. The simulator provides a test bed for automated driving, driving assistance, collision avoidance and navigation strategies and tactics. Apart from providing a cost-effective way for the design and testing of concepts, the knowledge gained through the simulation may potentially be adopted by the vehicle automated driving system as rules when navigating in the real world. Our case study shows that the simulator is indeed a powerful tool for analysis and assessment of algorithm performance.

1. INTRODUCTION

Road vehicles nowadays have many automated features incorporated, for example, the automatic gear transmission and anti-lock braking, just to name a few. Ideally speaking, the ultimate of vehicle automation must be the autonomous driving and navigation of the vehicle. In this respect, most of the ITS programs in the world have included research and development in areas such as the automated highway system (AHS), (ITS-America[1], ITS-Japan[2]), anti-collision autonomous support (T-TAP[3]), autonomous intelligent cruise control and route guidance (PROMOTE[3]). In these programs, they share a common goal of reducing drivers' burden of driving, improving safety and transport efficiency. To achieve the ultimate, a multitude of advanced technologies must be integrated into the vehicle to enable an intelligent and informed decision to be made. These technologies include vision, wireless communications, sensors (radar, ultrasonic, magnetic, etc.), control devices, warning devices, information storage, visualization, and of course, an information fusion framework for integration. The impact of automated driving is that accidents may possibly be avoided when better driving and obstacle avoidance techniques are employed. As a result, transport efficiency may also be improved[2].

Perhaps, to further enhance the prospect of automated driving, the research achievements and lessons learnt in Autonomous Mobile Vehicles (AMV) should not be overlooked. AMV such as the JPL Planetary Rover[4] and the Navlab[5] of CMU, use visual and non-visual sensors for signal acquisition, and artificial neural nets and fuzzy logic for control, path planning and navigation[6]. Although most of the AMV development are for indoor applications, there are also a few outdoor cases where an unstructured environment is assumed[7]. Besides, there is an increasing amount of activities in learning and studying AMV characteristics and behaviors through simulation[8]. This computer-aided approach provides a more flexible platform for programming, testing and debugging. A common practice in AMV simulation includes path planning and collision avoidance in a world model. Such simulations are often crude, one-off and custom-made, while experimentation indoor or even outdoor on a limited basis is still possible, although not desirable. However, physical experimentation of road vehicles is more restrictive, expensive and the tolerance for error is much lower. Therefore, the use of simulation and visualization tools at the onset should be considered as an essential and integral part of developing automated driving strategies and methodologies.

In this paper, an integrated virtual world simulator for road vehicles and networks is presented. The structure of the simulator is modular and object-oriented, where the virtual world is hierarchically constructed from three classes of objects: active dynamic, passive dynamic and static. It supports 2D/3D real-time graphic rendering of the environment, vehicles, sensors and other objects which can be visualized on multiple X-windows. The simulator also supports a direct 'plug-and-play' of algorithms written in C/C++. Currently, a number of algorithms have been implemented and

0-7803-4269-0/97/$10.00

tested in the simulator including navigation, collision avoidance and path planning. The principle of the simulator is that road and vehicle information and their associated static or dynamic objects are made to reflect the real world environment, where automated driving, driving assistance, collision avoidance and navigation strategies can be tested. Apart from providing a cost-effective way for the design and testing of concepts, the knowledge gained may potentially be adopted by the real vehicle automated driving system as rules when navigating in the real world. Our case study shows that the simulator is indeed a powerful tool for analyzing and assessing the performance of various techniques, in addition to the ability of being able to visualize the progress of the event.

The organization of this paper is as follows: Section 2 overviews the simulator platform; Section 3 describes the graphic rendering aspects of the simulator; Section 4 gives detailed explanation of the virtual world, vehicle, sensor and object models; Section 5 outlines the action decision module; Section 6 presents a case study and depicts results of the study; and this paper is concluded in Section 7.

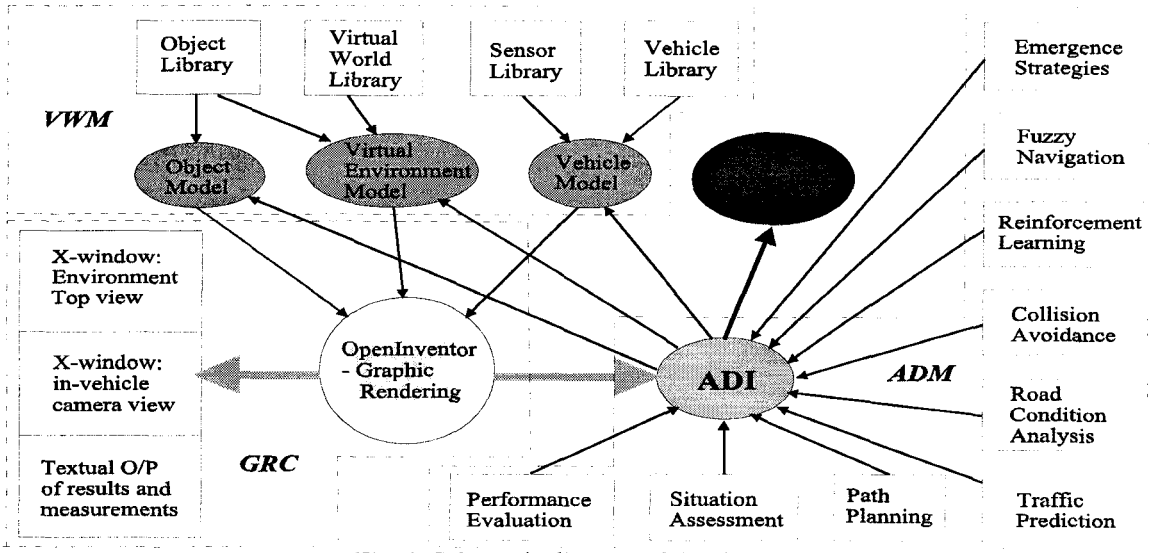

Fig. 1 Schematic diagram of the simulator

2. SIMULATION PLATFORM

2.1 Methodology

At the system level, the simulator is divided into three modules: the Graphic Rendering and Control Module (GRC), the Virtual World Module (VWM) and the Action Decision Module (ADM). The GRC supports the 2D/3D real-time rendering of graphic and textual display on multiple X-windows, and the control of the simulation sequence. The role of the VWM module is to assemble the virtual world and to ensure the correctness of the created objects. The correctly created object or world is represented by three levels of abstraction. The first level is where all the available details of the virtual world is described and represented. The second level is a simplified model of the first level for coarse localization and estimation of motion. The third level represents a 2D view of the virtual world map for path and contingency planning purposes. This three-tier hierarchy gives a degree of granularity according to the type of actions required. The role of the ADM is to provide the decisions planned for the dynamic objects such as vehicles in the virtual world at each time step. These decisions are made where the actions are executed and displayed on the GRC. The sensor inputs, camera views and etc. are then returned to the ADM for decisions in the next time step. Using this methodology, real-time simulation of the event as it happens becomes possible and the viability of the decision functions that derived the actions can also be studied and assessed. As multiple sensors including visual and non-visual are considered in the execution, the knowledge acquired in the ADM can be used as rules and guidelines in real-world vehicle navigation.

2.2 Simulation Architecture

The architecture of the simulator is depicted in Fig. 1. The GRC is developed on the SGI IRIX® OS and *OpenInventor*® platform where 2D/3D graphic objects, light source and view ports can be defined and rendered in real-time[9,10]. The input to the GRC is the virtual world, the objects and the vehicles. For static objects

such as buildings, their location in the virtual world is predefined. For mobile objects, initial default locations are chosen, and they can be changed or rearranged interactively through the GRC. The output of the GRC are first, the in-vehicle camera view and bird-eye's view which can be displayed on two X-windows simultaneously. These views are updated in real-time to enable a realistic viewing of the event. Textual results of the paths and measurements can also be output for detailed analysis. Second, the GRC also outputs location information and sensor information such as the distance values from the in-vehicle radar to the Action Decision Interchange (ADI) in the ADM.

The VWM is compiled in two formats: the virtual world is in **.iv* whereas the objects, sensors and vehicles are in *OpenInventor* scene graphs. It consists of four libraries: virtual world, vehicles, sensors and objects. The virtual world library contains the major road network of Hong Kong divided into regions. The vehicle library contains three vehicle types currently: car, bus and truck, and the sensor library contains a camera, ultrasonic model and a radar model. The object library contains fixtures such as building, lamp post, dust bin and human being. From the four libraries, objects can be invoked and placed inside the virtual world at fixed or default locations, and sensors can be invoked and combined with a vehicle to form one complex entity. This allows the GRC to treat them as one description. The scene and **.iv* descriptions form the basic model for the GRC to render and display.

The ADM consists of an ADI and a set of decision algorithms. The purpose of the ADI is to interchange decisions by the algorithm in execution and the actions that has been taken by the GRC. These activities take place at every time step during the simulation. The ADI is designed for interchanging decisions and actions with the real vehicle in the future. At present, a path planning algorithm, a fuzzy-based navigation algorithm, and an neural network-based reinforcement learning algorithm have been implemented. As long as an algorithm is written in C/C++, it can be directly *plug* into the ADI and tested.

3. GRAPHIC RENDERING & CONTROL

3.1 *OpenInventor* Features

OpenInventor is an object-oriented 3D graphics tool builds on top of *OpenGL* which supports a *Scene Description Language*, the C/C++ programming and an **.iv* ASCII file format. It is designed to permit users to focus on creating scene objects that can be collected in a scene database for viewpoint-independent rendering. Its programming model reduces programming time and extends 3D programming capabilities. Together with a rich set of objects already defined in *OpenInventor*, scenes can be easily created and used[10].

In principle, a scene is a hierarchical collection of nodes and associated attributes. The nodes are basically objects that represent 3D shape, property, or group. Once a scene graph is created, it can be attached to a scene database which is a collection of one or more scene graphs and objects. Inventor programs create or read their own copies of scene database each time they execute.

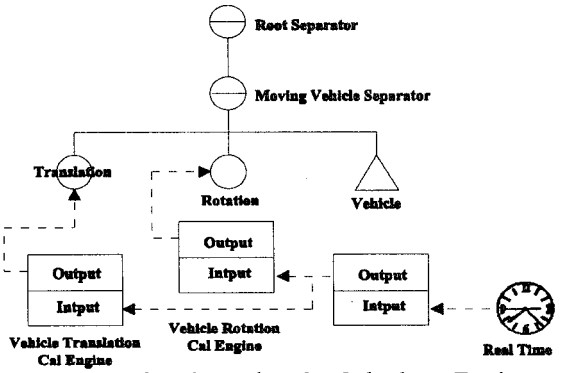

Fig. 2: Animation using the Calculator Engine

In *OpenInventor*, there is a defined class of objects called *Engine* to enable behavior animation through data sensors, timer sensors and behavioral engines which can be connected to automatically animate elements of the scene graph. By connecting the *Calculation Engine* to the *Translation Node* and *Orientation Node* of an object, the object can be moved around in the scene. For example, Fig. 2 illustrates the scene graph for the calculator engine animating the moving vehicle.

3.2 Camera and Bird-eye's Views

The in-vehicle camera view is designed for realistic visualization. Apart from real-time 3D rendering capability, key-entry and display of position and orientation information have also been included. As far as the display of the vehicle's position and orientation is concerned, a field sensor is attached to the position field of the viewer's camera. This sensor then responds to the changes in the attached field by invoking the user-supplied callback function. This is accomplished by the following functions:

```
SoCamera *camera = myviewer -> getCamera();
SoFieldSensor *mySensor =
new SoFieldSensor(cameraChangedCB,camera);
mySensor -> attach(&camera->position);
```

It then displays the information to the user through the use of Text nodes. The node graph of the in-vehicle camera view is depicted in Fig. 3. The bird-eye's view is generated using the same method, except that no key-entry function is incorporated. As both views are displaying the same event, the two views must be synchronized when performing the rendering. This is achieved by using the Motif-compliant form widget to lay both graphics rendering components inside the same X-window.

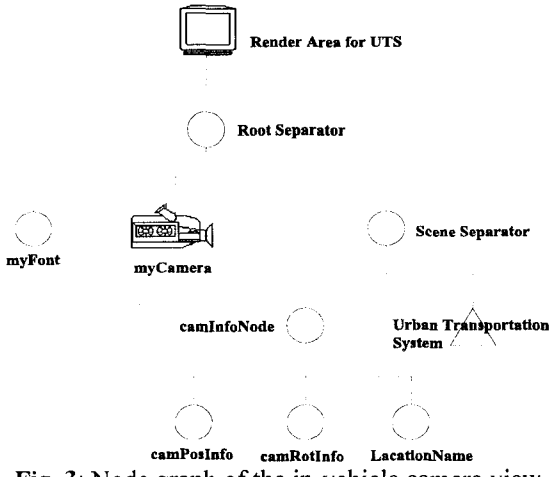

Fig. 3: Node graph of the in-vehicle camera view

4. VIRTUAL WORLD MODELING

4.1 Environment Model

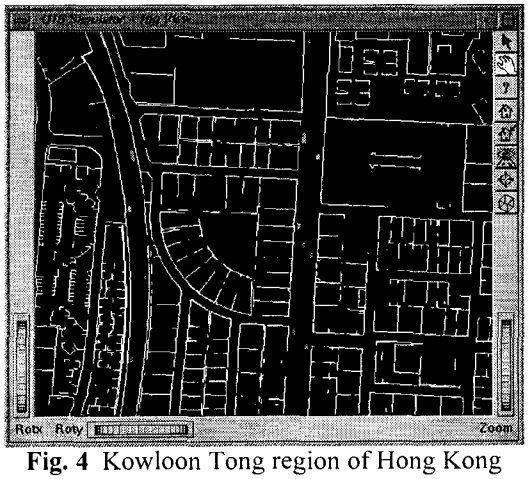

Fig. 4 Kowloon Tong region of Hong Kong

The transportation network used in this simulator is a digital road map of Hong Kong comprising three main areas: the Hong Kong Island, Kowloon and New Territories. The complete network can be divided into a number of districts or regions. As these road maps are designed for 2D viewing in the first place, building height was introduced to create a 3D view and perception for the two views. These maps were then complied into the Virtual World Library and can be invoked for building a complex virtual world. Fig. 4 depicts the Kowloon Tong region of the digital road map and Fig. 5 shows its bird-eye's view.

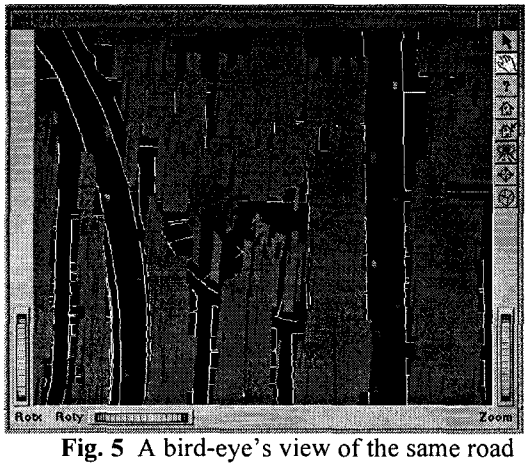

Fig. 5 A bird-eye's view of the same road

4.2 Object Model

In this simulation, all the 3D objects in the virtual world are made of primitive shapes such as cubes, spheres and cones. The modeling can be easily done by scene graphs while the *Scene Description Language* provides a convenient way of translating and combining these primitive objects. For instance, two human beings have been modeled in this way (Fig. 6). In order for these objects to be integrated with the virtual environment, the dimensions of the objects must be measured in the same SI unit as that of the environment, i.e., all lengths are measured in meters. Further, the integration is performed by inserting the separator nodes of the objects to the road map and then translating them to the correct location.

Fig.6 Female and male modeled in the simulator

4.3 Vehicle and Sensor Model

The modeling of the vehicle is similar to the static objects, with the exceptions that each vehicle has a list of attributes such as maximum velocity, acceleration/ deceleration rate, fuel consumption, among others, attached to it. When a vehicle is inserted into the virtual world, it's associated attributes are made available to

the simulator automatically. Moreover, the insertion of a vehicle into the virtual world is different from the static objects because of the dynamic nature of the vehicle. In order to make it moves in the virtual world, the behavior animation using timer sensors and engines must be used. The scene graph describing how this is done is depicted in Fig. 2.

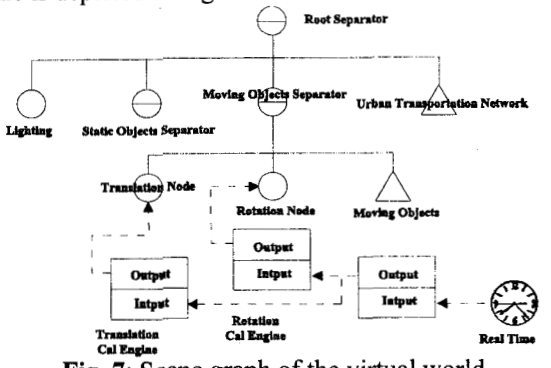

Fig. 7: Scene graph of the virtual world

The sensor models considered for the simulator are classified by their function as dead reckoning sensors, heading sensor, obstacle detection sensors and other positioning sensors. In addition to these, two visual sensors, the perspective camera and orthographic camera were also modeled. To model a vehicle with sensing capabilities, a selection of these sensors are first combined with the vehicle. This combined model is treated as one model when integrated into the virtual world. A typical virtual world scene graph is depicted in Fig. 7.

5. ACTION AND DECISION

The ADI plays two roles, of which the first one is to convert the decisions made by an algorithm into the corresponding *OpenInventor* commands that are subsequently implemented in the virtual world. Second, it also converts the sensor input from the virtual world into relevant data that can be passed back to the algorithm for making its future decisions. For instance, let us assume one of the vehicle (A) in the virtual world is employing a fuzzy logic collision avoidance algorithm for it's tactical navigation. At each time step, distance estimation of objects in the vicinity of vehicle A in the virtual world is determined by the on-board radar sensors. These distance measurements are then transmitted from the GRC through the ADI to the collision avoidance algorithm. Based on the information, the collision avoidance algorithm makes a further trajectory estimation for vehicle A for the next time step, which is then transmitted via the ADI back to the *Calculator Engine* in the simulator to enable the engine to calculate the correct translation and heading angle of vehicle A in the virtual world. The same ADI mechanism is employed for all the algorithms supported in the ADM.

6. CASE STUDY

Overtaking is a common undertaking for road vehicles. In this case study, we assume that the overtaking sequence is performed on the two lanes of the four-lane Waterloo Road in Kowloon Tong of Hong Kong where the other two lanes are for on-coming traffic. The other vehicles using the same two lanes are assumed to be maintaining constant speed, while the vehicle to perform overtaking is able to accelerate, decelerate and change its steering angle. For the simulation, the navigation algorithm and obstacle avoidance algorithm are used as the decision functions, where the vehicle concerned is equipped with radar sensor models for distance estimation, and an in-vehicle camera model for viewing. The distance of the vehicle from the objects or other vehicles on the road are estimated and sent to the algorithms via the ADI. When a safe overtaking is possible, the sensor space is mapped into the vehicle action space by the fuzzy inference and defuzzification. The new trajectory information (new velocity and steering angle) is then released to the ADI, which maps it to the virtual world actions. Fig. 8 depicts the in-vehicle camera views of the overtaking sequence in the virtual world. Fig. 8(a) shows the camera view just before the overtaking. Fig. 8(b) shows the camera view just after the vehicle has committed the overtaking action. Fig. 8(c) shows the camera view while overtaking and Fig. 8(d) shows the camera view after the overtaking action has been completed.

7. CONCLUSION

In conclusion, a fully integrated and realistic virtual world simulation platform for studying road vehicle navigation has been developed. It offers real-time graphic rendering of the virtual world and visualization of events and supports direct plug-and-play of algorithms written in C/C++ in the simulator. The case study of using an actual region of the Hong Kong digital road map proves that the simulation can effective be used for its intended purpose. Further in-depth study of the overtaking strategies and tactics will be conducted using the same set of parameters, to consider the effect of varying the speed of the other vehicles and allowing multiple overtaking sequences to occur simultaneously by different vehicles. The knowledge gain through the simulation is believed be important for the actual automated navigation of real road vehicles.

ACKNOWLEDGEMENT

This research is supported by the HKU/CRCG grant#337-062-0016

REFERENCES

1. "Intelligent Transportation Systems Projects", *Department of Transport*, United States of America, Jan 1996.
2. "Comprehensive plan for ITS in Japan", *Government of Japan*, Japan, July 1996.
3. "ITS Handbook in Japan", Ministry of Construction, *Highway Industry Development Organization*, Japan, 1996
4. U. Rembold & P. Levi, "Sensors and control for autonomous robots", *Intelligent Autonomous Systems*, 1987, Section 4.
5. C. Thorpe, M. H. Herbert, T. Kanade & S. A. Shafer, "Vision and navigation for the Carnegie-Mellon Navlab", *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 10, No.3, May 1988, pp.241.
6. N. H. C. Yung & W. K. Tsang, "On the current state-of-the-art research in autonomous mobile vehicles/robots", Research Report #EEE95001-HCY/WKT, Department of EEE, *The University of Hong Kong*, 1995.
7. H. Ishiguro, K. Nishikawa & H. Mori, "Mobile robot navigation by visual sign patterns existing in outdoor environemnt", *Proc. of IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, July, 1992, pp. 636-641.
8. K. Kimoto & S. Yuta, "A simulator for programming the behavior of an autonomous sensor-based mobile robot", *Proc. of the 1992 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, July, 1992, pp.1431-1438.
9. Josie Wernecke, Open Inventor architecture Group, "The Inventor Mentor", ISBN 0-21-62495-8, *Addison Wesley Publishing Co.*, 1993.
10. Josie Wernecke, Open Inventor Architecture Group, "The Inventor Toolmaker", ISBN 0-201-62493-1, *Addison Wesley Publishing Co.*, 1994.

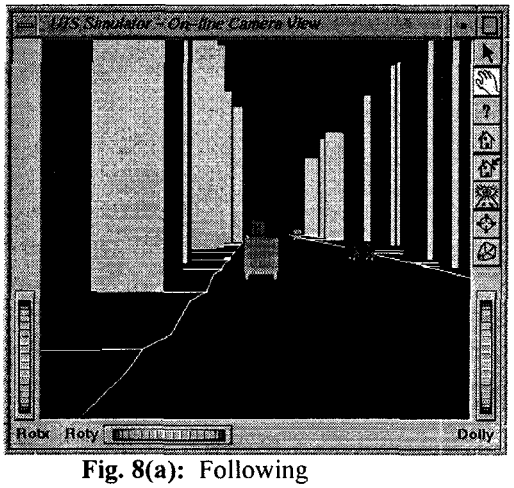

Fig. 8(a): Following

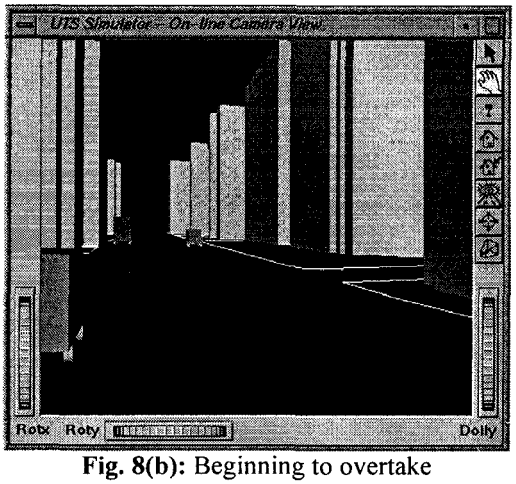

Fig. 8(b): Beginning to overtake

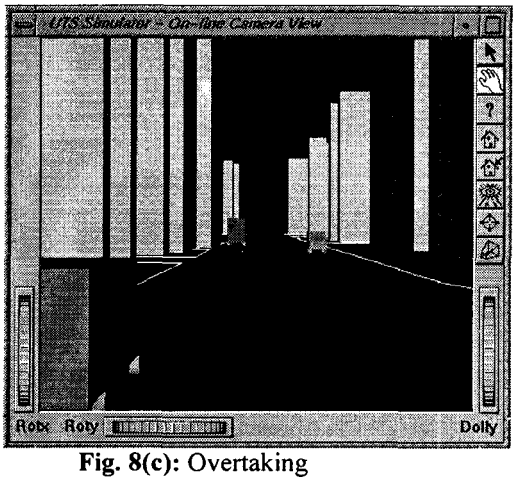

Fig. 8(c): Overtaking

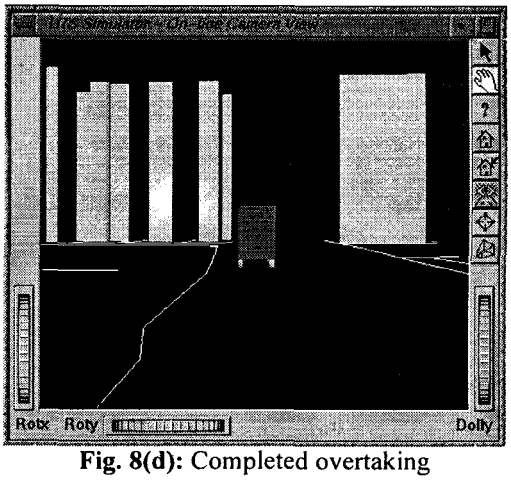

Fig. 8(d): Completed overtaking