EXPECTATIONS - An Autonomous Mobile Vehicle Simulator

N. H. C. Yung and C. Ye
Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong

ABSTRACT

This paper describes a fully integrated mobile vehicle simulator - *EXPECTATIONS*. The structure of the simulator is one of modular and object-oriented, where the virtual environment, static and dynamic objects and their interactions are hierarchically constructed. It supports 2D/3D real-time graphic rendering of the composite environment, which can be visualized on multiple X-windows in a time synchronized manner, in which vehicle or object movement can be animated in accordance with the calculation of the algorithms written in C/C++. Algorithms such as path planning, behavior learning, collision avoidance and navigation strategies can be 'plug-and-play' easily through the so called Action Decision Interchange concept. Apart from providing a realistic visualization tool for AMV development, it also supports fast algorithmic study and development, and the knowledge learnt through the simulation may potentially be used by the physical vehicle in real operations.

1. INTRODUCTION

Autonomous Mobile Vehicle (AMV) research over the past two decades has seen rapid advancement in both theory and applications of AMV in areas ranging from flexible manufacturing, fire fighting to highway automated navigation. Since the late 60's to early 70's, many prototypes have been developed, for example, the JPL Planetary Rover [1], Navlab & Neuro-Nav [2,3] of CMU, MELDOG of MITI [4], KAMRO [5], of which visual and non-visual sensors were used for signal acquisition; artificial neural nets and fuzzy logic were used for control, path planning and navigation [6]. Apart from a small number of outdoor cases [7], most of the AMV development have been for indoor applications.

Concurrently, there is an increasing amount of activities in learning and studying AMV characteristics and behaviors through simulation [8,9] which aims at widen the scope of AMV research beyond physical experimentation. This computer-aided approach provides a more flexible platform for AMV programming, testing and debugging. A common practice in AMV simulation is to perform path planning [10] and collision avoidance [11,12] in an environment model. However, such simulations are often crude, one-off and custom-made. As physical experiments are more restrictive, expensive with lower error margin; and it is unrepeatable in some case, the use of simulation and visualization tools at the onset should be considered as an essential and integral part of AMV development. An obvious choice here is the Virtual Reality (VR) technology. From the point of AMV navigation, this technology provides possibilities for several applications, such as using Virtual Environment (VE) for off-line knowledge acquisition, learning /training, teaching and planning. Undoubtedly, the VR technology in AMV is one of the most timely and exciting research areas. However, applying VR in navigation studies is not without problems. For instance, real AMV and sensor behaviors are often extremely complex which are hard to be modeled accurately; and real environment consists of numerous objects having high degree of variability. In order to reap the benefit of VR, we must establish our understanding of how VE knowledge relate to real world environment and how this man-made and idealized knowledge can be interpreted and used by the physical AMV when tackling real world navigation.

This paper describes a fully integrated mobile vehicle simulator - *EXPECTATIONS*. The structure of the simulator is one of modular and object-oriented, where the virtual environment, static and dynamic objects and their interactions are hierarchically constructed. It supports 2D/3D real-time graphic rendering of the composite environment including tables, chairs, walls, humans, other AMV and etc., which can be visualized on multiple X-windows in a time synchronized manner, in which vehicle or object movement can be animated in accordance with the calculation of the algorithms written in C/C++. Algorithms such as path planning, behavior learning, collision avoidance and navigation strategies can be 'plug-and-play' easily through the so called Action Decision Interchange concept. Apart from providing a realistic visualization tool for AMV development, it also supports fast algorithmic study and development, and the knowledge learnt through the simulation may potentially be used by the physical vehicle in real operations.

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 explains how the virtual environment, vehicle, sensors and objects are modeled; Section 5 outlines the action decision interchange concept and how it works; Section 6 presents a case study and depicts results of the study; and this paper is concluded in Section 7.

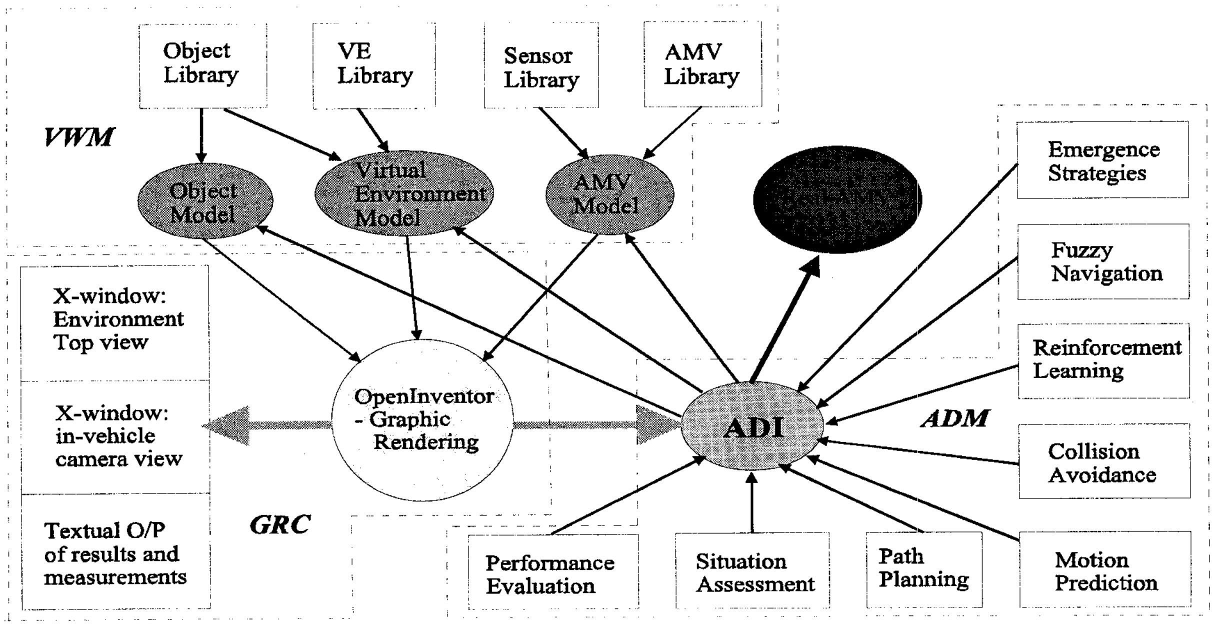


Fig. 1 Schematic diagram of the simulator

2. SYSTEM OVERVIEW

The architecture of the simulation platform is depicted in Fig. 1. The simulator is developed on the *SGI IRIX*® *OS* and *OpenInventor*® (Trademarks of Silicon Graphics Inc.) platform. There are three modules at the system level: Graphic Rendering and Control Module (GRC), Virtual World Module (VWM) and Action Decision Module (ADM). The GRC supports the 2D/3D real-time rendering of the VE, camera views and textual display on time synchronous multiple X-windows, and the control of the simulation sequence.

The role of the VWM module is to assemble the virtual world from the VE and available objects, and to ensure the correctness of the created world. In essence, the environment is represented by three levels of abstraction. The first level is where all the available details of the VE 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. For instance, a third level map is often sufficient for path planning, where speed is critical. A second level representation is good for more detail route estimation, where visualization and object identification are best performed at the first level.

The role of the ADM is to provide the decisions made for the dynamic objects in the VE at each time step. These decisions are translated via the Action Decision Interchange (ADI) into actions for the GRC. The sensor inputs, camera views and etc. are then returned via the ADI for further decisions. Using this methodology, real-time simulation and visualization of the event as it happens becomes possible and the performance of decision functions may be studied and assessed. As physical AMV may be modeled and it's actions simulated, knowledge acquired through this concept may be used as rules and guidelines in real-world AMV navigation.

3. GRAPHIC RENDERING & CONTROL

OpenInventor is an object-oriented 3D graphics tool built on top of *OpenGL* which supports a *Scene Description Language*, C/C++ 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 based on the scene database reduces programming time and extends 3D programming capabilities. Together with a rich set of objects already defined in *OpenInventor*, scenes can be easily created [13].

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 depicts a scene graph in the simulator.

To perform behavior animation, the *OpenInventor Engine* object class can be used. Using the behavioral engines with data sensors and time sensors, animation objects may be created. By connecting the *Calculation Engine* to the *Translation Node* and *Orientation Node* of an object, the object can be moved around in the scene.

To enable visualization of events, *OpenInventor* also supports the creation of multiple view points. In *EXPECTATIONS*, a number of view points can be specified. The usual two are the AMV camera view and the 2D plan view. Besides the real-time 3D rendering capability, key-entry and display of position and orientation information can also be displayed. As far as this 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 and obtains the camera position and orientation. It then displays the information through the use of Text nodes. The node graph of the AMV camera view is depicted in Fig. 3.

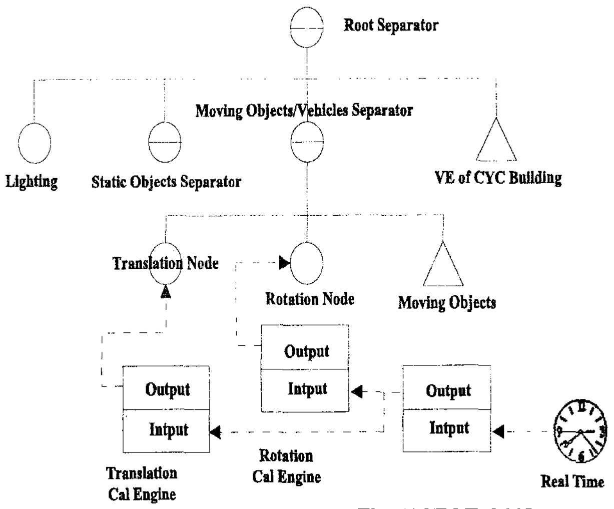


Fig. 2 A VE scene graph in *EXPECTATIONS*

The top view of the VE is generated using the same method, except that the orthographic projection is used. As both views are displaying the same event, the two views must be time 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. This allows both views to be synchronized and displaying the same event.

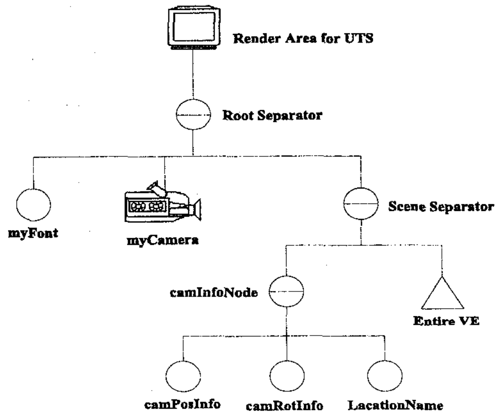


Fig. 3 Node graph of the AMV camera

4. VIRTUAL ENVIRONMENT

4.1 Environment Model

The VE used in this simulator is the floor plan of the CYC Building of our university. This floor plan file was originally in 2D where a third dimension (height) was introduced to create a 3D VE for AMV view. A number of different VE were then complied into the *Virtual Environment Library* and can be invoked separately, or in some case, for building a more complex one. Fig. 4 depicts two views of the 5th floor VE mentioned above.

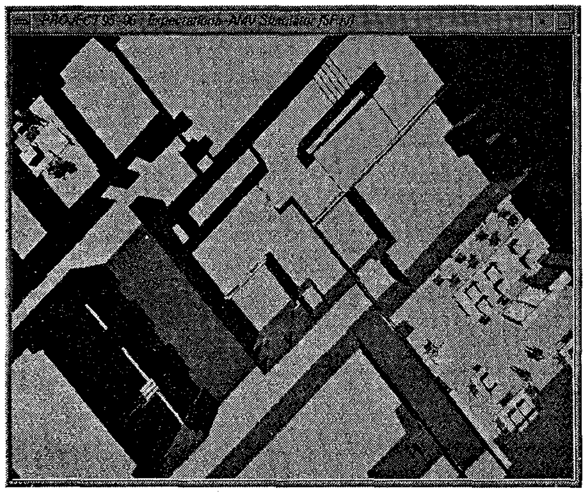


Fig. 4 (a) Virtual environment of the CYC 5th floor

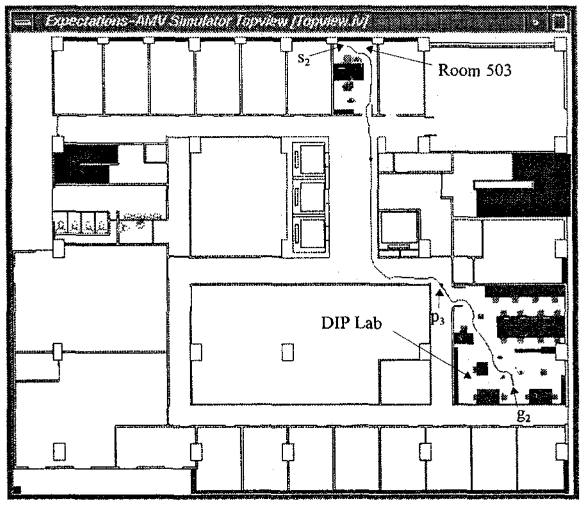


Fig. 4(b) Plan view of the CYC 5th floor

4.2 Object Model

The 3D objects in the simulator are created from primitive shapes such as cubes, spheres and cones which are described by scene graphs. This includes the static objects such as chair, table, bookshelf and computer; and dynamic objects such as robot, AMV and human. These objects are collected in *Object Library* together. In order for these objects to be integrated with the VE, the dimensions of the objects must be measured in the same SI unit as that of the environment and of the same ratio, that is, all lengths are measured in millimeters. Further, the integration is performed by inserting the separator nodes of the objects to the VE and then translating them to the correct location. Some of these objects are depicted in Fig. 5.

Fig. 5 Object models

4.3 Vehicle and Sensor Model

The AMV models are basically the same as the static object models except that each vehicle has a list of attributes such as maximum velocity, maximum acceleration/deceleration, among others, attached to it. When a vehicle is inserted into the VE, it's associated attributes are made available to the simulator. Moreover, the insertion of a vehicle into the VE is different from the static objects because of the dynamic nature of the AMV. In order to make it moves in the virtual world, the behavior animation using timer sensors and calculator engines must be used. Fig. 6 depicts an AMV with a camera at the top and light sources at the top and sides and three wheels. The scene graph for animating the moving of the AMV is given in Fig. 7.

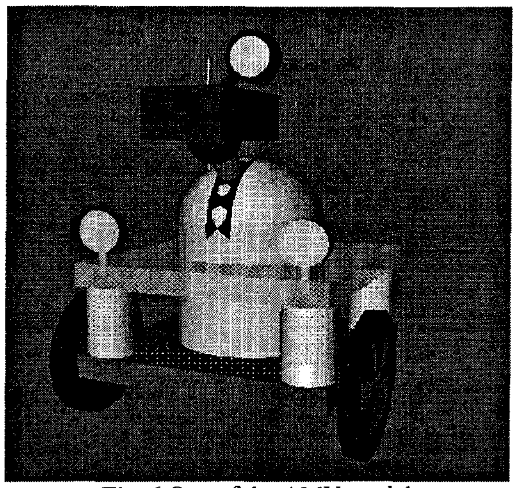


Fig. 6 One of the AMV model

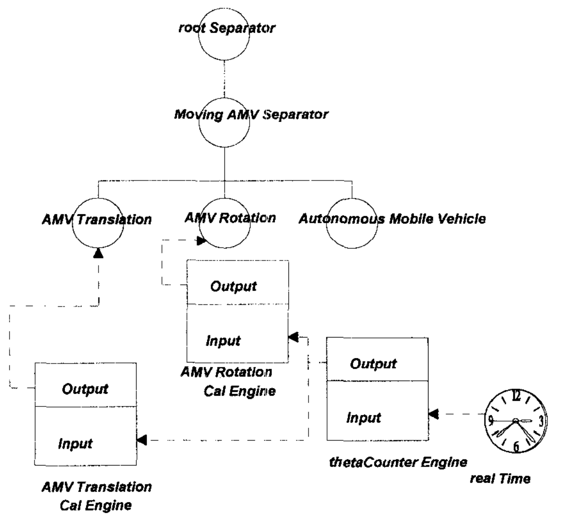


Fig. 7 Calculator engine for AMV animation

The sensor models considered for the simulator are classified by their function as dead reckoning sensors, such as odometer, optical encoder; heading sensor such as gyroscope and geomagnetic sensors; and obstacle detection sensors such as ultrasonic sensor, laser and radar. 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 VE. A camera view of the VE is depicted in Fig. 8.

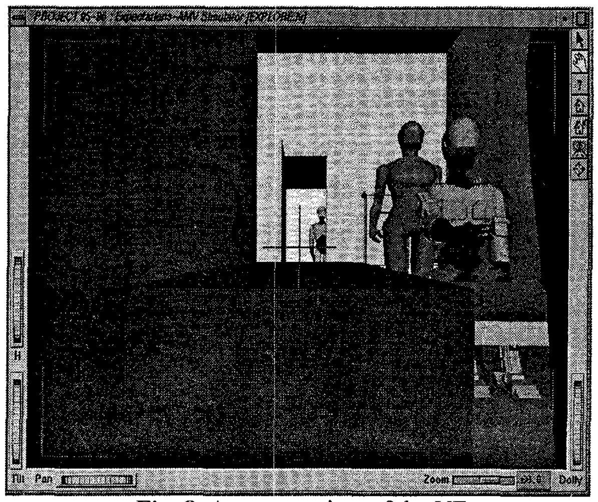


Fig. 8 A camera view of the VE

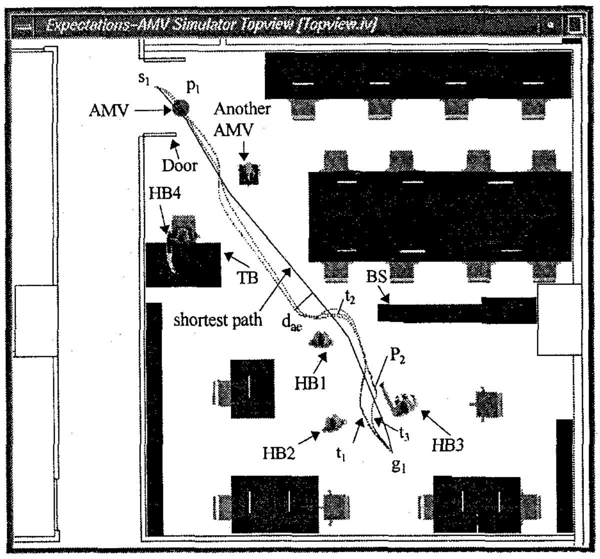
5. ACTION AND DECISION

The ADM supports two functions: interchange of actions and decisions between the GRC and the algorithm making the decisions; and the algorithms. 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 VE. Second, it also converts the sensor inputs from the VE into relevant data structures that can be passed back to the algorithm for making its future decisions.

For instance, let us assume one of the AMV-A in the VE is employing a fuzzy logic collision avoidance algorithm for its tactical navigation. At each time step, distance measurements of objects in the vicinity of AMV-A are taken by the ultrasonic sensors equipped on AMV-A. These distance measurements are then transmitted from the GRC through the ADI to the collision avoidance algorithm written in C. Based on this information, the collision avoidance algorithm makes the action decision for AMV-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 orientation of AMV-A in the VE. The same ADI mechanism is employed for all the algorithms supported in the ADM.

6. NAVIGATOR IN THE VE

In this case study, the neural/fuzzy based navigator proposed in [14] was adopted to demonstrate how the proposed simulator can be used as a platform for verifying and tuning navigation algorithm as well as visualizing the navigation event in the VE.



Dotted line–path determine by different navigation algorithms
Solid line–shortest path determine by Visibility Graph Method

Fig. 9 Navigation in the DIP lab

The VE is chosen to be the DIP lab on the 5th floor of the CYC Building. Static objects such as tables, chairs, computers and bookshelves in the VE were arranged as one would for a laboratory. Five dynamic objects apart from the operating AMV (A) were inserted into the VE randomly with a robot sitting on a chair, three humans at the open space, and another AMV (B) parked behind the robot. The zoom-in plan view of the VE is depicted in Fig. 9. A navigation task can be specified by entering via the mouse or keyboard the start location (& heading angle) and goal location. After the task was defined, the navigator began with the acquisition of ultrasonic sensor distance measurements via the ADI. Based on the measurements, the navigator made fuzzy decision on the appropriate actions. These values were converted into *OpenInventor* commands that moved AMV-A towards the goal. This action-decision process continued until the goal was reached. From start to finish, one X-window showed the plan view of the VE and all its contents, while another X-window displayed the camera view of AMV-A. As these two windows were time synchronized, the action-decision sequence may be studied carefully. For instance, the speed of the navigation, acceleration/ deceleration, steering angle, path calculated and obstacle avoidance actions can all be studied as the event unfolds. In addition, the paths of using different navigation algorithms may also be recorded and compared with an ideal path. The visualization of the event from AMV-A camera is shown in Fig. 10 where (a) depicts the view just before entering the lab; (b) depicts the view at p_1 and (c) depicts the view midway during the navigation.

Due to the large amount of information provided by the ***EXPECTATIONS*** simulator, the navigation algorithm was tuned and refined, and performance analysis was conducted by changing the number of obstacles (objects) inside the laboratory and by comparing the paths determined by different algorithms, or under different set of parameters. As a result, the navigator achieves better performance and adaptability.

7. CONCLUSION

In conclusion, a fully integrated and realistic virtual world simulation platform — ***EXPECTATIONS*** has been developed. Basic on the *OpenInventor* platform, it offers real-time graphic rendering of the virtual world and visualization of events and supports direct 'plug-and-play' of navigation and decision algorithms written in C/C++. This structure enables fast algorithm development and tuning, as well as detailed study of the dynamics and behaviors of AMV in known or unknown environments. Besides, the rules and knowledge derived from the simulator can potentially be used for navigation purpose in the real world. The case study of navigating in an indoor VE proves that the simulation methodology can effective be used for its intended purpose.

Future directions of this research includes first, extending the VE and object library; second, refining the VE and object descriptions for more realism; third, incorporating more algorithms to test out the capability of the simulation; and fourth, structuring the knowledge and information through the simulation which may be readily used by a physical AMV.

8. REFERENCES

[1] U. Rembold & P. Levi, "Sensors and control for autonomous robots", *Intelligent Autonomous Systems*, 1987, Section 4.

[2] 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, pp. 362-373, May 1988.

[3] M. Meng & A. C. Kak, "NEURO-NAV: a neural network based architecture for vision-guided mobil robot navigation using non-metrical models of the environment", Proc. Of the 1993 IEEE/RSJ International Conf. On Intelligent Robots and Systems, pp.750-757, 1993.

[4] S. Tachi & K. Komoriya, "Guide dog robot", International Symposium on Robotics Research, Vol.2, MIT press, pp.333-340.

[5] J. Zhang & P Bohner, "A fuzzy control approach for executing subgoal guided motion of a mobile robot in a partially-known environment", Proc. Of the 1993 IEEE/RSJ International Conf. On Intelligent Robots and Systems, pp.545-550, 1993.

[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 Electrical & Electronic Engineering, *The University of Hong Kong*, 1995.

[7] H. Ishiguro, K. Nishikawa & H. Mori, "Mobile robot navigation by visual sign patterns existing in outdoor environment", *Proc. of the 1992 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, 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*, pp. 1431-1438.

[9] T. Skewis and V. Lumelsky, "Simulation of Sensor-Based Robot and Human Motion Planning", *Proc. of the 1992 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, pp. 933-940.

[10] C. K. Choi et al, "Dynamic Path-Planning Algorithm of a Mobile Robot Using Chaotic Neuron Model", *Proc. of the 1995 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems,* pp. 456-461.

[11] A. Zelinsky et al, "Using Augmentable Resource to Robustly and Purposefully Navigate a Robot", *Proc. of 1995 IEEE Int. Conf. On Robotics and Automation.* pp. 2586-2592.

[12] B. J. A. Kröse and J. W. M. Van Dam, "Adaptive State Space Quantization for Reinforcement Learning of Collision-free Navigation", *Proc. of the 1992 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, pp. 1327-1332.

[13] Josie Wernecke, Open Inventor architecture Group, "The Inventor Mentor", ISBN 0-21-62495-8, *Addison Wesley Publishing Co.*, 1993.

[14] N. H. C. Yung and C. Ye, "An intelligent navigator for mobile vehicles", Proc. of International Conference on Neural Information Processing, pp. 948-953, 1996.

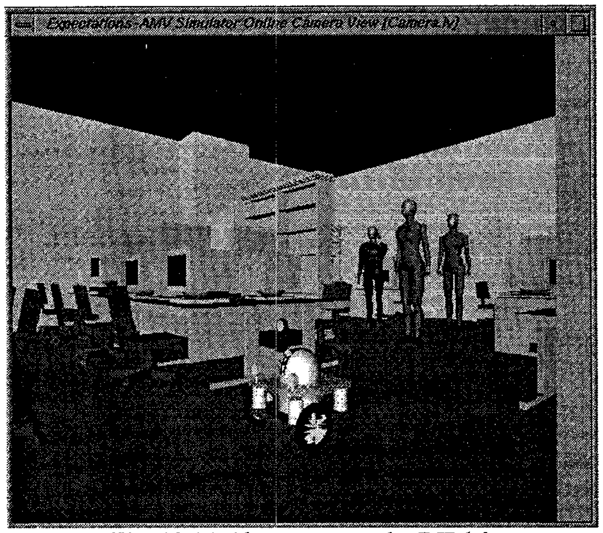


Fig. 10 (a) About to enter the DIP lab

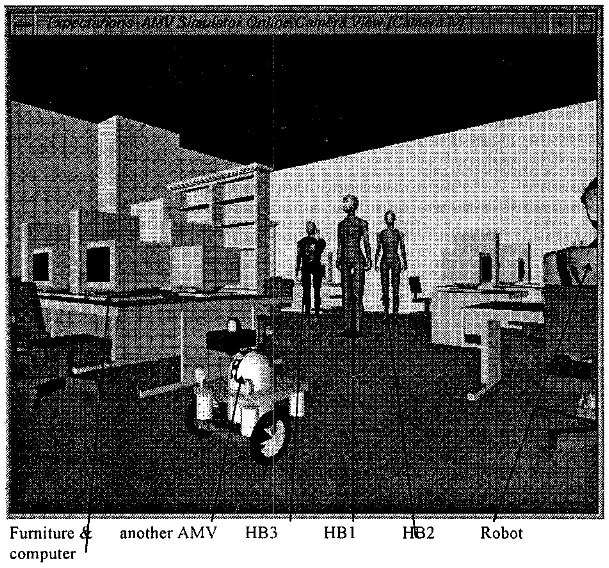


Fig. 10 (b) Camera view at p_1

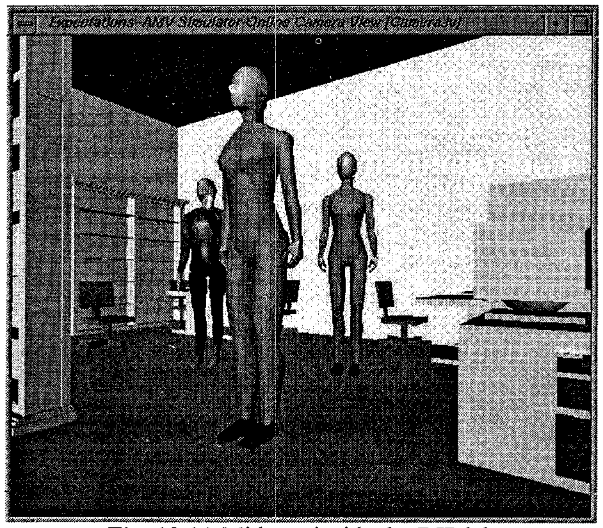


Fig. 10 (c) Midway inside the DIP lab