A MOBILE ROBOT FOR CORRIDOR NAVIGATION: A MULTI-AGENT APPROACH

Y. Ono, H. Uchiyama, W. Potter
Artificial Intelligence Center
University of Georgia
Athens, GA
yukiono@uga.edu

ABSTRACT

This project focuses on building an autonomous vehicle as the test bed for the future development of an intelligent wheelchair, by proposing a framework for designing and implementing a mobile robot control program that is easily expandable and portable to other robotic platforms. Using a robot equipped with a minimal set of sensors such as a camera and infrared sensors, our multi-agent based control system is built to tackle various problems encountered during corridor navigation. The control system consists of four agents: an agent responsible for handling sensor inputs, an agent which identifies a corridor using machine vision techniques, an agent which avoids collisions by applying fuzzy logic decision making to proximity data, and an agent responsible for locomotion. In the experiments, the robot's performance demonstrates the feasibility of a multi-agent approach.

Categories and Subject Descriptors

I.2.9 Robotics---*Autonomous vehicles, Commercial robots and applications,* I.2.11 Distributed Artificial Intelligence---*Multiagent systems*, I.4.6 Segmentation--*Edge and feature detection*, I.2.3 Deduction and Theorem Proving---*Uncertainty, "fuzzy," and probabilistic reasoning*.

General Terms

Design, Experimentation, Standardization.

Keywords

Multi-agent systems, Collision avoidance, Fuzzy logic controller, Machine vision, Commercial robots and applications.

1. INTRODUCTION

Recent advances robotics technologies have already made enormous contributions in many industrial areas. There are numerous robotic applications found in a variety of areas in our society such as surveillance systems, quality control systems, AGVs (autonomous guided vehicles), and cleaning machines [9, 10]. Robots are now expected to become the next generation of rehabilitation assistants for elderly and disabled people. One of the more researched areas in assistive technology is the development of intelligent wheelchairs. By integrating intelligence into a powered wheelchair, a robotic wheelchair has the ability to safely transport a user to their desired destination. Our project was originally inspired by the need for robotic assistance. It hopefully will lead to future study on building an intelligent powered wheelchair which ultimately assists disabled people with navigation.

In order to achieve successful navigation in a narrow hallway, a robot must exhibit fundamental abilities such as recognizing a corridor and detecting and avoiding collisions. An autonomous robot equipped with agents performing such tasks requires information about the environment where the robot is situated. The robot used in our project was developed using a minimal set of sensors such as a camera and infrared sensors. The camera captures the front view of the surrounding environment, and the infrared sensors detect objects in the nearby vicinity. The employed agents are capable of handling uncertainty by compensating for the inaccuracy of the sensor data using a-priori knowledge and heuristic methods. The corridor navigation agent, for example, processes a captured image and identifies a corridor using machine vision techniques, and the collision detection agent avoids walls and obstacles by using fuzzy logic to interpret the sensor inputs.

Due to recent developments within the robotics industry, building a robot has become much more effortless with the aid of commercially available robot kits. Using these kits allows us to possibly reuse the robot and the robot control program since the kit is usually provided with useful development tools. Reusing a program which has already been tested and proven to perform certain tasks (at least under certain conditions) can save enormous time and cost in building and testing a robot. This project focuses on building an autonomous vehicle as the test bed for future development of an intelligent wheelchair, by proposing a framework for designing and implementing a mobile robot control program that is easily expandable and portable to other robotic platforms.

2. RELATED WORK

Many engineering maneuvers have been developed to resolve navigation problems for powered wheelchairs. While the issues in assistive technology have been researched in great depth, there are only a few intelligent wheelchairs commercially available for end users. Examples include the Smart Wheelchair[1] (designed only for children) which protects the user from collisions, and navigates him/her from room to room following the tracks made

ACMSE '04, April 2-3, 2004, Huntsville, Alabama, USA.

[1] The CALL Centre (University of Edinburgh). More information available at: http://callcentre.education.ed.ac.uk/

with reflective tape on the floor, and SCAD from Chailey Heritage[2] which provides specific controlled actions to aid users in certain situations [7].

Most intelligent wheelchairs are, on the other hand, still under development or only sold to schools and institutes for research purposes. One of the early intelligent wheelchairs was built by Yanco who introduced Wheelesley, a robotic wheelchair system [11]. This semi-autonomous robot travels safely in an indoor environment using various sensors. Using the graphical interface, users can easily navigate the wheelchair by selecting a simple instruction which represents a course of several navigational tasks. NavChair is one of the most successful intelligent wheelchairs. It was developed at the University of Michigan [5]. The tasks of this robotic wheelchair consist of the following three modes: (1) obstacle avoidance, (2) door passage, and (3) wall following. The control system automatically changes the mode according to the environmental surroundings. The TAO series of wheelchairs developed by Applied AI Systems Inc. are famous intelligent wheelchairs for exploring in an indoor environment [6]. In addition to the tasks performed by the NavChair, TAO-1 and TAO-2 also have two additional tasks: (1) escape from a crowded environment and (2) perform landmark based navigation. Currently, TAO-7 is in use.

At the KISS Institute for Practical Robotics, TinMan II is in the early stages of assistive robotics development [7]. The TAO and TinMan series have been sold to other institutions such as the MIT AI Lab (Wheelesley) and the University of Rochester as prototypes for intelligent wheelchair development [12]. Rolland (the Bremen Autonomous Wheelchair) assists the user in obstacle avoidance and door navigation [4]. MAid (Mobility Aid for Elderly and Disabled People) was experimented with in crowded environments (e.g., a railway station) and successfully navigated in heavy passenger traffic [8].

3. HARDWARE

3.1 Robot Kit

The hardware used in this project is a commercial robot kit called the ER1 Personal Robot System, supplied by evolution robotics™. The robot kit includes the control software, aluminum beams and plastic connectors to build a chassis, two assembled nonholonomic scooter wheels powered by two stepper motors, one 360 degree rotating caster wheel, a power module, a battery (12V 5.4A), and a web-camera. Our experimental robot also carries additional accessories, nine infrared sensors and extra beams and connectors for reinforcement. A laptop computer, Dell™ Latitude C640 (Intel® Mobile Pentium® 4 processor 2.0GHz with 512 MB RAM), is used as a controller device, and Windows XP Professional is loaded as the operating system.

Figure 1. Customized ER1

The hardware of the ER1 robot kit empowers users to customize the robot for their objectives. The reconfigurable chassis enables us to design a purposive mobile robot, and the extensions (extra cameras, sensors and grippers) can be easily added to the system if necessary. The purpose of this experiment is to build a robot as a test-bed for a future wheelchair project, so the autonomous robot is modeled after the typical powered wheelchair with two independent wheels.

3.2 Sensors

In this experiment, nine infrared (IR) sensors and a single web camera are used and gather information about the environment. Figure 2 depicts the arrangement of sensors installed on the robot. The camera, *Logitech® QuickCam® Pro 4000*, is mounted in front of the vehicle capturing the front view. The 160 x 120 32-bit RGB image is updated and saved in memory at the rate of 10 frames per second. The camera is connected to the PC through a USB (Universal Serial Bus) port and used mainly for recognizing a path in the hallway. The IR sensors (*evolution robotics IR sensor pack*) enclose the rectangular robot fairly evenly for 360° of coverage as shown in Figure 2. According to the sensor specifications, objects within a distance between 15 cm and 100 cm should reliably be acquired in reasonable ambient lighting conditions. Behaviors such as collision detection and obstacle avoidance are designed to perform actions based on the information given by these sensors. Behaviors are extensively discussed in the next section.

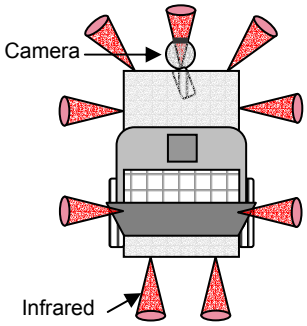


Figure 2. Sensor Arrangement

4. APPROACH

4.1 Incremental Design

The ultimate goal of our robotic experiments is to build a controller which can be used in the future study of a robotic wheelchair. In order to build such a robust and compatible controller, we must build the program as a complete system with a set of complete behaviors, which enables the robot to be tested in a real world environment. Rodney A. Brooks at the MIT AI Laboratory suggested in his famous article that building complex robots (he calls creatures) which coexist in the world with humans must be incrementally built in the same manner as biological evolution [1]. For instance, a single-cell amoeba which wanders the world without any goal and a human who exhibits intelligent behaviors are both complete biological systems although there is a difference in the degree of intelligence. During the astronomical time span, biological evolution on earth started from the lower intelligence of amebas and now has ended up with human level intelligence up to this point. Brooks' idea of building a robot mimics the process of evolution. The concept of this incremental design helps the entire project of building an intelligent system to advance toward the goal steadily one step at a time.

[2] Chailey Heritage Clinical Services: http://www.southdowns.nhs.uk/directory/chailey/

4.2 Architecture

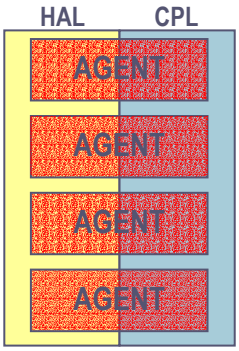


Figure 3. System Architecture

The robotic system architecture used in this project consists of two layers which constitute four task-oriented agents. Each agent is composed of two functional layers, the Hardware Layer and the Component Layer. The Hardware Layer is a collection of modules communicating with the robot's hardware devices such as the camera, infrared sensors and motors. The Hardware Layer is implemented using Visual C++ .NET since the ER1 kit comes with a development environment that specifies the language. The SDK (Software Development Kit) already contains libraries to help in accessing the hardware components of the robot, which reduces the amount of redundant effort by developers. This layer functions as a bridge between the upper-level layer and the hardware. The Component Layer contains the intermediate functional modules which constitute the higher-level behaviors as agents. One module can be shared by two or more agents, which reduces redundancy in coding. The Component Layer is implemented with Java™ Technology (Sun Microsystems, Inc). Programming in Java has enormous advantages in building a robot control application. The next section explains why.

4.3 Platform Independence

Recent advancements in computer technology enable us to build a Java-based control program which can solve problems in real-time. Although it is still slower than native programs, it does not bother us if the program completes the job in a reasonable time. Java provides more distinctive strength in controlling a mobile robot system. Many robot control programs in the robotics literature were hardware specific mainly because of the uniqueness of the hardware, operating systems limits on the programming language, and the development environment specific to the platforms. Meanwhile, platform independence, one of the most attractive features of Java, allows the control program to operate similar robots on various computer platforms and operating systems with minor modifications.

Having this feature in the robot control program, we can test the program on a smaller scale prototype. Using a test-bed for a robot control program is especially needed if the robot is a large intelligent system equipped with many features. Another advantage of using Java in mobile robot control is the availability of APIs (Application Program Interface). Sun Microsystems already provides a number of useful tools for free which are unavailable in C/C++. In the meantime, many research institutions, companies, and even individuals distribute miscellaneous APIs with or without charge. In this project, several useful APIs are used in the control program. For example, the fuzzy collision detection agent uses the NRC FuzzyJ Toolkit freely distributed by the National Research Council of Canada. This fuzzy toolkit API provides the capability of handling fuzzy concepts and reasoning. Using these APIs usually maintains system compatibility.

5. SOFTWARE

5.1 Multi-Agent System

The goal of this project is to design and implement a naive but robust and easily expandable robot control package that is portable to heterogeneous system and hardware platforms, starting with a commercial robot kit as a test bed. Figure 4 depicts the simplified diagram representing the multi-agent system using a blackboard. The agents basically interact with the other components of the system by manipulating information on the blackboard. The blackboard operates as a central repository for all shared information and a communication medium for all agents. The information on the blackboard may represent facts, assumptions, and deductions made by the system during the course of solving a problem. An agent is a partial problem solver which may employ a different problem-solving strategy and try to contribute to the solution by viewing the information on the blackboard. The system has four independent agents: Fuzzy Collision Detector, Corridor Recognizer, Sensor Handler, and Drive Controller. Note that the arrows in Figure 4 represent information flow. The diagram shows that all four agents are allowed to read / write information on the blackboard. Each one

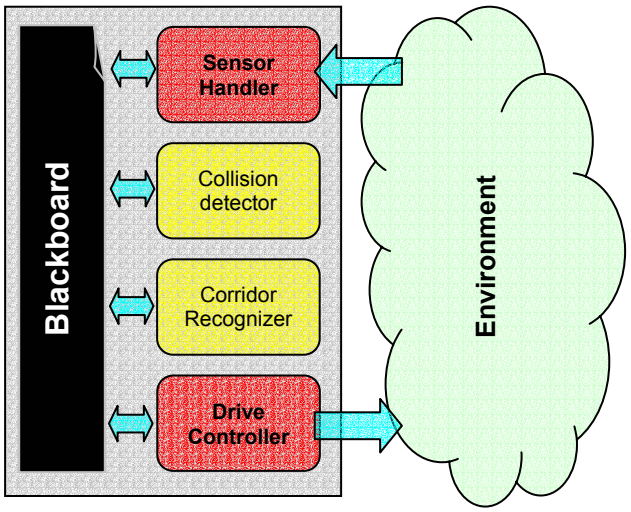


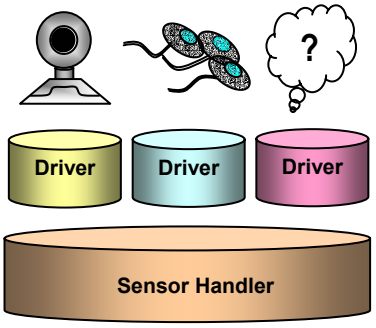
Figure 4. Multi-agent System with Blackboard

of the four agents basically executes their tasks independently using information on the blackboard and posts any result back to the blackboard.

5.2 Agents

The four agents (Sensor Handler, Collision Detector, Corridor Recognizer, and Drive Controller) and the blackboard make up the control system. The agents are classified into two groups. The Sensor Handler and the Drive Controller belong to the first group, which has access to and interacts with the environment (see Figure 4). The other two, the Fuzzy Collision Detector and the Corridor Recognizer are strictly prohibited from having direct access to the environment. As a result, they perform tasks based on the information acquired from the blackboard. There is no global controller for these agents, and each of them independently tries to make a contribution to the system during the course of navigation. The agents that directly interact with the environment are designed for the purpose of incrementally adding more tools to the

system. Particularly, the Sensor Handler is responsible for all the sensors installed on the robot, and maintains the ability to deal with the sensors without any extensive impact on system modification and configuration.



Sensor Handler is capable of integrating multiple sensors via the drivers.

Figure 5. Sensor Handling Apparatus

As we discussed in Subsection 4.2, the agents are composed of two layers, the Hardware Layer and the Component Layer. The Sensor Handler specially benefits from this layered architecture by having device drivers at the Hardware Layer (Figure 5). Layering the drivers between the Component Layer and the physical sensor apparatus, the Sensor Handler can maintain its adaptability by having task-oriented modules at the Component Layer which only deal with the symbolic representation of sensor data (e.g., digitized frequencies or an array of pixels). Each sensor requires a driver written in a native programming language such as C or C++ which runs under the operating system (e.g., *.exe for Windows). Although the drivers can be freely designed and implemented, they are required to receive input commands and return output character strings by the Sensor Handler. This standardized I/O specification facilitates the implementation process of the agent.

The Drive Controller is also the one that has access to the environment. The agent primarily holds responsibility for the robot's actuator via the device driver that controls motors through the stepper control module. The Drive Controller monitors the blackboard and uses the navigational information for locomotion. With the layered framework, the Drive Controller has the same advantage as the Sensor Handler in its simple and structured implementation. For example, the agent is made of modules responsible for the motor initialization and termination, the

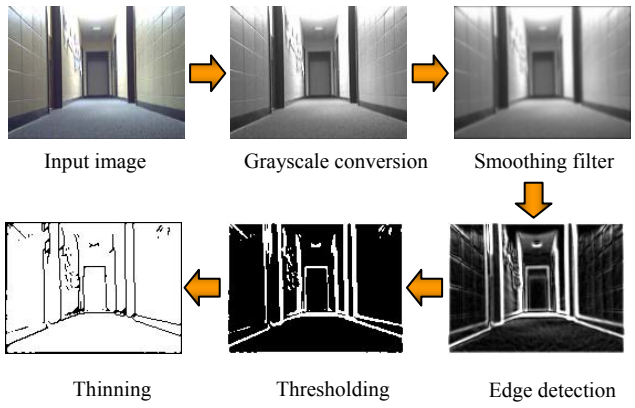


Figure 6. Flow of Low-level Image Processing

communication between layers, and the maneuvering of the robot. A decent number of motion parameters such as velocity, acceleration, turn-angle, straight distance, and driving duration are arranged for achieving flexible movements.

5.3 Corridor Recognition

The Corridor Recognizer agent is used to find the geometric features of an indoor environment from an input image and classify the image into the following three categories: corridor, wall, and obstacle. The agent roughly consists of two levels of image processing modules. The low-level image processing basically involves tasks regarding image segmentation. Figure 6 shows each step of the segmentation process. The JPEG image is acquired from the camera at the resolution of 160 x 120 pixels with a 32-bit (ARGB) color model. The image is then converted to grayscale (8-bit) for ease of computation while the pixel intensity values are stored in an integer array. Applying a Gaussian filter reduces noise in the image by blurring neighboring pixels and helps the edge detector to select correct edges. A Sobel edge detector is applied to the smoothed image. The grayscale colors in the image are reduced to black and white using the adaptive thresholding operator so as to remove unwanted details before applying a thinning operation. The threshold value is dynamically selected by performing a statistical analysis on the sampled pixel intensity values. Because of the nature of the Sobel operator, the thinning operator must be applied to reduce the lines with several pixels width to a single pixel width.

Results

Corridor: Yes

Wall: No

Obstacle: No

Figure 7. Corridor Recognition

After the lower-level image processing, the Hough transform with a-priori knowledge (constraints on the geometry features of a corridor) is used for extracting the line segments of a corridor path (Figure 7). The Hough transform possibly extracts all line segments in the image. In order to select lines which best represent the hallway, we need to use knowledge about corridors. The selection process involves two steps, elimination and verification. In the elimination phase, the lines whose slope does not fit the geometry constraint are thrown out first. Next, each line is compared with the edge maps. In this step, the pixels of a line matching the corresponding edge points are counted, and only lines with the matching pixels that go above a certain threshold are selected. In the verification step, after selecting the corridor path, the corridor recognition agent double-checks the lines to see if they really represent the hallway or not by performing a complete histogram analysis by comparing the image with the typical patterns of the histograms of images representing the environment (corridors, walls, and obstacle).

5.4 Fuzzy-Based Collision Avoidance

The agent called the Fuzzy Collision Detector is a fuzzy-based collision avoidance controller responsible for the safety of the robot used in this experiment. The fuzzy logic controller has one input fuzzy set for the sensor value and three output fuzzy sets for linear-distance, velocity and turn-angle. Each set is defined by one or more membership functions that map numeric values onto linguistic terms. The membership functions of the sensor value fuzzy set are shown in Figure 8. The fuzzy-based agent is fed

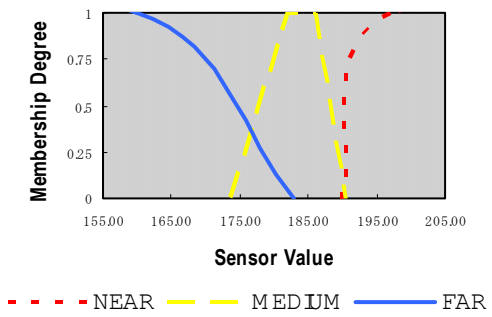


Figure 8. Membership Function

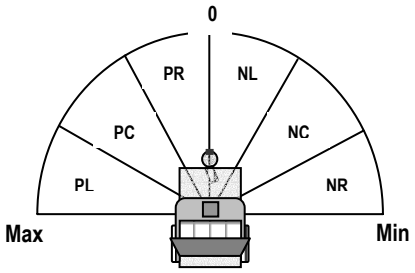


Figure 9. Turn-angle Fuzzy Set

with sensor values as input, acquired from a set of infrared proximity detectors. The values are fuzzified with designated linguistic terms (near, medium, and far). Among three output fuzzy sets, the turn-angle fuzzy set has been uniquely defined. The angle lies between -30°and 30° act as a default (adjustable via the navigation software interface). The total angle of 60° is divided into six amplitudes represented by six member functions, and each of which is associated with the following linguistic terms: positive-left (PL), negative-left (NL), positive-center (PC), negative-center (NC), positive-right (PR), and negative-right (NR). Figure 9 projects the six linguistic terms on the partitioned turn-angle fuzzy set. This scheme has already been established and

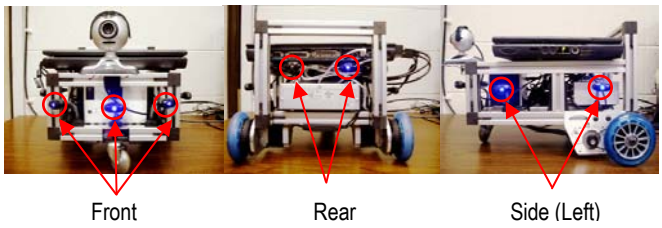


Figure 10. Sensor Arrangements

proven to be an effective method [2]. Once the input is fuzzified and all the output fuzzy sets are defined by appropriate linguistic terms, the fuzzy inference engine looks for a match between the input and the outputs.

The agent has 17 fuzzy rules in total: seven rules for the front sensors, two for the rear sensors, and four for each of the left and right sensors as shown in Table 1. The nine IR sensors are used as inputs, three sensors in the front, two in the back, and two on each side. The sensor arrangement is shown in Figure 10. Right columns in the table are the sets of conclusions. Notice that the rules do not necessarily carry the same set of conclusions. One may wonder if the rules sufficiently cover all possible collision circumstances which may demand more than seventeen rules. As a matter of fact, the number of inferences we can draw from these 17 rules is at most 192 combinations, and the linguistic output explosively multiplies on to numerical values in the defuzzification process. Defuzzification is the last step of the fuzzy logic controller. The outputs from each rule firing are combined using a fuzzy union operator, and the crisp output value is defuzzified by computing a centroid of the area that is enclosed by the output member functions.

Table 1. Fuzzy Rules

Variables are enclosed by [ ] and linguistic terms are capitalized. The antecedents are the sensor readings, and the conclusions are the robot's steering parameters.

	Antecedents (IF)	Conclusions (THEN)
1	[Front Center] is NEAR	[Distance] is BACKWARD [Velocity] is MEDIUM
2	[Front Center] is MEDIUM	[Distance] is ZERO [Velocity] is SLOW
3	[Front Center] is FAR	[Distance] is FORWARD [Velocity] is FAST
4	[Front Left] is NEAR	[Distance] is BACKWARD [Velocity] is MEDIUM
5	[Front Left] is MEDIUM	[Turn Angle] is NL
6	[Front Right] is NEAR	[Distance] is BACKWARD [Velocity] is MEDIUM
7	[Front Right] is MEDIUM	[Turn Angle] is PR
8	[Rear Left] is NEAR	[Distance] is ZERO [Velocity] is SLOW
9	[Rear Right] is NEAR	[Distance] is ZERO [Velocity] is SLOW
10	[Left Front] is NEAR [Left Rear] is NEAR	[Turn Angle] is NL
11	[Left Front] is NEAR [Left Rear] is MEDIUM	[Turn Angle] is NC
12	[Left Front] is NEAR [Left Rear] is FAR	[Turn Angle] is NR
13	[Left Front] is MEDIUM [Left Rear] is FAR	[Turn Angle] is NL
14	[Right Front] is NEAR [Right Rear] is NEAR	[Turn Angle] is PR
15	[Right Front] is NEAR [Right Rear] is MEDIUM	[Turn Angle] is PC
16	[Right Front] is NEAR [Right Rear] is FAR	[Turn Angle] is PL
17	[Right Front] is MEDIUM [Right Rear] is FAR	[Turn Angle] is PR

6. EXPERIMENTAL RESULTS

The experiments are conducted on a narrow straight corridor in an office-like indoor environment with a relatively dimmed lighting

condition. As a result, the robot has shown both desired and problematic behaviors. In the matter of collision detection, the robot was able to avoid collisions with obstacles and walls. The fuzzy-based collision detection agent maneuvered the vehicle around and navigated it to the end of the hallway without collisions. There were also some problems, however. First of all, the sensor agent was often distracted by ambient light, which caused the retardation in navigating the robot. The second problem is the advisability of the fuzzy rules. The agent sometimes makes magnified conclusions about the turning angle in large amplitude, which results in zigzag locomotion. The corridor recognition agent analyzes the visual input and the contents of the front view. Although the agent has made correct decisions on recognizing a corridor most of the time, the robot has shown some problematic behavior. In principle, the robot control system has no central brain in the system, and any information posted on the blackboard must be handled and interpreted by each agent. A subtle timing difference in the results of agents sometimes causes destructive behaviors. While the agents are completely independent and perform tasks in parallel, one agent occasionally delays in updating the blackboard. The delays are most likely instantaneous and unnoticeable but sometimes come in between two states where the robot is making a significant transition. For example, the robot was facing along the corridor and then turned to the wall in the next ten milliseconds. At this moment, the blackboard could contain contradictory facts about the environment such as "going straight with a moderate speed" (the message from the collision detection agent) and "facing a wall" (the message from the corridor recognition agent).

As mentioned in the previous sections, the objective of this experiment was to design and build a robot control program that is independent of the system platform and easy to expand (modify) for future study. To begin with, the control system succeeded in facilitating modularity and usability. The complete modulation in the multi-agent architecture brought forth an effortless implementation, and the intelligible user interface has navigational parameters that are all adjustable, and realizes smooth experiment processes. Meanwhile, some problems are found regarding system stability. During navigation, the control system often lost control because of a system failure. The system must be integrated with a reliable fail-safe procedure.

7. DISCUSSION

The vision-based agent, the fuzzy-based agent, and the agents responsible for hardware components all cooperate within the blackboard-based multi-agent system. The robot and the control system were presented and analyzed in the experiments. Although the robot did not exhibit the perfectly desired performance, the multi-agent approach in the design criteria has proved its feasibility in mobile robot navigation. The proposed layered architecture enabled the control system to be easily expandable, and the use of Java technology made the system independent of operating systems and robot hardware. In the current ongoing research, a module has been added to the blackboard agent which synchronizes the results and resolves the conflicts among the updated information. Further study is sought to design an agent that will perform landmark-based navigation, extend the machine vision techniques, and learn via an agent with a neuro-fuzzy controller for learning an environment so that no manual calibration is necessary.

8. REFERENCES

[1] Brooks, R. A. 1991. Intelligent without representation. *Artificial Intelligence* 47: 139-59.

[2] Fayad, C. and Webb, P. 1999. Optimized fuzzy Logic based on algorithm for a mobile robot collision avoidance in an unknown environment. *Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany.
http://www.cs.nott.ac.uk/~cxf/Papers/Optimised_Fuzzy_Logic.pdf

[3] Gomi, T. and Griffith, A. 1998. Developing intelligent wheelchairs for the handicapped. In Mittal et al. eds., *Assistive technology and AI*. LNAI-1458, Berlin: Springer-Verlag, 150-78.

[4] Lankenau, A., Röfer, T. and Krieg-Bruckner, B. 2003. Self-localization in large-scale environments for the Bremen Autonomous Wheelchair. In Freksa and et al. eds., *Spatial Cognition III*. LNAI-2685. Berlin: Springer-Verlag, 34-61.

[5] Levine, S.P. and et al. 1999. The NavChair assistive wheelchair navigation System. *IEEE Transactions on Rehabilitation Engineering* 7(4): 443-51.

[6] Miller, D. 1998. Assistive robotics: an overview. In Mittal et al. eds., *Assistive technology and AI*. LNAI-1458. Berlin: Springer-Verlag, 126-36.

[7] Nisbet, P. D. 2002. Who's intelligent? Wheelchair, driver or both? *Proceedings of the 2002 IEEE International Conference on Control Applications*, Anchorage, AK, 760-5.

[8] Prassler. E. and et al. 2001. A robotic wheelchair for crowded public environments. *IEEE Robotics and Automation Magazine* 8(1): 38-45.

[9] Trahanias, P.E. and et al. 1997. Navigational support for robotic wheelchair platforms: an approach that combines vision and range sensors. *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, NM, 1265-70.

[10] Wijesoma, W. S., Khaw, P. P., and Teoh, E. K. 2001. Sensor modeling and fusion for fuzzy navigation of an AGV. *International Journal of Robotics and Automation* 16(1): 14-25.

[11] Yanco, H. A. and et al. 1995. Initial report on Wheelesley: a robotic wheelchair system. *Proceedings of the Workshop on Developing AI Applications for the Disabled*, held at the International Joint Conference on Artificial Intelligence, Montreal, Canada.
http://www.cs.uml.edu/~holly/papers/ijcai95.pdf

[12] Yanco, H. A. 1998. Integrating robotic research: a survey of robotic wheelchair development. *AAAI Spring Symposium on Integrating Robotic Research*, Stanford, CA.
http://www.cs.uml.edu/~holly/papers/sss98.pdf