Osaka University Knowledge Archive

Title	Research on Control Architecture and Visual Information Processing for Mobile Robots
Author(s)	渡邊, 睦
Citation	大阪大学, 1997, 博士論文
Version Type	VoR
URL	https://doi.org/10.11501/3132592
rights	
Note	

Osaka University Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

Osaka University

Research on Control Architecture and Visual Information Processing for Mobile Robots
（移動ロボットの制御構成と視覚情報処理に関する研究）

Mutsumi Watanabe

May 1997

Research on Control Architecture and Visual Information Processing for Mobile Robots

by

Mutsumi Watanabe

Submitted to the Department of Computer-Controlled

Mechanical Systems

for the degree of

Doctor of Engineering

at

OSAKA UNIVERSITY

on May 1997

©Mutsumi Watanabe, 1997

Research on Control Architecture and Visual Information Processing for Mobile Robots

Mutsumi Watanabe

ABSTRACT

This thesis describes research and development concerning an architecture of a driving controller and vision information processing methods to generate basic behaviors for autonomous mobile robots working in unstructured dynamic environments, such as offices, homes, nuclear power plants and streets.
To counter various kinds of failures which may occur in a dynamic environment, a new behavior-based architecture is proposed to realize both efficiency in attaining a robot's mission and robustness against failures.
In addition, computer vision algorithms to realize sophisticated obstacle avoidance ability are presented.
That is, a collision-free space detection method by processing stereo images, and a moving obstacle detection and recognition method by processing optical flow information acquired from dynamic images, are explained.
A small cart-type robot has been developed for navigation experiments in indoor environments. Experimental results in real scenes have demonstrated the effectiveness of the architecture and these methods.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to Professor Yoshiaki Shirai for his insights, academic advice and encouragement.
I am also grateful to Associate Professor Yoshinori Kuno for his kind guidance. The research related in Chapters 2 and 3 was developed under his guidance.
It is my pleasure to mention the following superiors of mine at Toshiba Corp., who have significantly influenced my career and research:

Dr.Masatsugu Kidode for his direct guidance and instructive advice; Mr.Haruo Asada, Mr.Sho Tsunekawa, Mr.Hiroshi Hoshino and Dr.Akio Okazaki for their direction; Dr.Teruhiko Ukita and Dr.Makoto Nakamura for valuable advice, and Mr.Ken-ichi Maeda for facilitating this thesis and his encouragement.
In working on this thesis, I also benefited from the generous assistance and cooperation of Mr.Kazunori Onoguchi, Dr.In So Kweon, Mr.Minoru Ishikawa, Mr.Yasukazu Okamoto, Mr.Nobuyuki Takeda and other colleagues. In particular, Mr.Onoguchi contributed to the technical content of Sections 2 and 3, especially Chapter 3.4. Dr.Kweon significantly contributed to the whole of Section 2. Mr.Takeda developed the algorithms presented in Chapter 4.2.2.
Dr.Charles Stephen Wiles advised me about both contents and English descriptions. Mr.Minoru Yagi helped with experiments on robot navigation by acting as a moving obstacle (shown in Fig.2.12).
Above all, the love and support of my wife, Yasuko, and my mother, Fumiko, contributed to the completion of this thesis. I will remain deeply in debt to them.

May 1997
Mutsumi Watanabe

Contents

ABSTRACT **1**

ACKNOWLEDGEMENTS **2**

Contents **3**

1 Introduction **5**
1.1 Development of research 5
1.1.1 Attainment of quick reflexive response for real-time navigation 9
1.1.2 Feasibility study of solutions to Artificial Intelligence (AI) problems 10
1.1.3 Pursuit of navigation performance 11
1.1.4 Summary of issues in mobile robot research . . . 15
1.2 The aim of this research 16
1.2.1 Basic behaviors for mobile robots 16
1.2.2 Desired control systems for mobile robots 18

2 Control architecture **21**
2.1 Related work 22
2.2 System design 24
2.2.1 Constitution of group-based multi-agent system . 26
2.2.2 Driving command determination method 27
2.3 Detailed explanation of agents 30
2.3.1 Energy-based motion determination algorithm . . 30
2.3.2 Reflexive Behavior Agent Group 35
2.3.3 Purposive Navigation Behavior Agent Group . . . 36

2.3.4 Adaptive Behavior Agent Group 37
2.4 Motion Executor 40
2.5 Summary of issues in the proposed architecture 43
2.6 Experimental results 44
2.6.1 Simulation experiments 44
2.6.2 Experiments on real robot navigation 46

3 Free space detection by stereo vision 53
3.1 Related work 54
3.1.1 Geometry of stereopsis methods 54
3.1.2 Related work on passive stereopsis methods . . . 54
3.1.3 Related work on generation of environment description 57
3.2 Disparity Prediction Stereopsis Method 58
3.2.1 Prediction of disparity and feature edge matching 59
3.2.2 Extraction of passage region 63
3.2.3 Measurement of obstacle position 65
3.3 Experimental results 67
3.3.1 Discussions 77
3.4 Planar Projection Stereopsis Method 85

4 Moving object recognition by monocular vision 89
4.1 Related work 90
4.2 Object recognition by optical flow analysis 95
4.2.1 Optical flow detection 97
4.2.2 Moving obstacle candidate detection 99
4.2.3 Moving object recognition 106
4.3 Experimental results 114
4.3.1 Discussions 124

5 Conclusion 127

Bibliography 131

LIST OF PUBLICATIONS AND TECHNICAL REPORTS BY THE AUTHOR 140

Chapter 1

Introduction

1.1 Development of research

Autonomous mobile robots have been extensively studied since the 1960's.

Conventional manufacturing robots in practical use generally have limited intelligence because they are engaged in fixed work in restricted environments such as factories. However, higher intelligence is required for an autonomous mobile robot able to cooperate with human beings in less restricted environments. To realize such robots, it is necessary to develop a robust and purposive architecture for robot navigation control and sensor information processing in dynamic environments.

In the 1960's and the 1970's, research on vision-based intelligent mobile robots largely consisted of conducting feasibility studies relating to the several key artificial intelligence (AI) problems, such as reasoning or image understanding in block worlds (see Fig.1.1). The researchers of intelligent mobile robots were mainly interested in indoor environments. Recently, with a view to improving the adaptability of robots, research has focused on learning issues. The learning of environments by automatic map generation based on vision-based observation has already been done[2] [3][4].

A type of vision sensor system (HyperOmni Vision) with a hyperboloidal mirror to obtain 360° angle of view range of images around a mobile robot has also been developed[5].

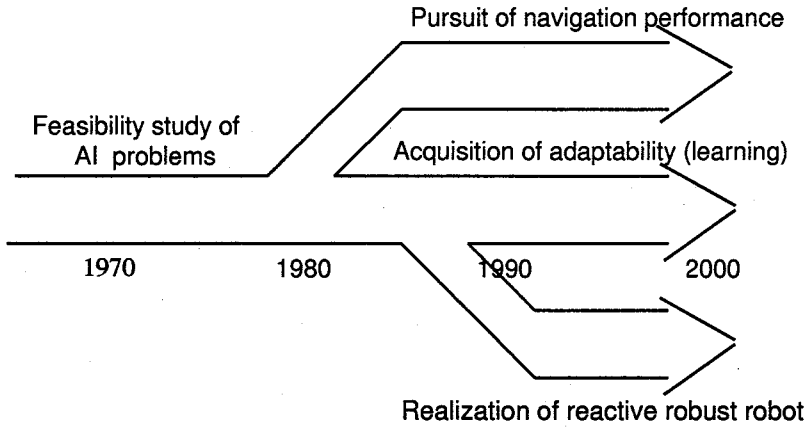


Figure 1.1: Development of vision-based intelligent mobile robot research.

With regard to the automatic learning of behaviors, vision-based reinforcement learning for acquiring purposive behavior has been studied[6].

In the reinforcement learning scheme, a robot and an environment are modeled by two synchronized finite state automatons interacting in discrete time cyclical processes. The robot senses the current state of the environment and selects an action. Based on the state and the action, the environment makes a transition to a new state and generates a reward that is passed back to the robot. Through these interactions, the robot learns a purposive behavior to achieve a given goal.

An example of this, in which a robot learns to shoot a ball into a goal, has been successfully achieved without knowledge of the 3D environment or its kinematics/dynamics.

In the 1980's, more complete mobile robots which moved in the outdoor world started to appear. Navigation environments were usually roads. Most of the robots were equipped with multiple sensors, such as ultrasonic ranging sensors, laser rangefinders and TV cameras, to improve the sensing ability for intelligent navigation. The main object of these robots was efficient navigation. In some cases, attempts were made to achieve real-time navigation by using a reactive function,

such as white-painted line tracking, in a restricted environment without obstacles.

These robots mentioned above had clear missions in terms of navigation (for example, reaching a final goal or making a map). The missions were distinctly given by the developer. That is, these robots were human substitutes.

In the middle of the 1980's, a novel type of reactive robot which had no clear mission appeared. The main task of these robots was to maintain safety while moving around an office environment, or in practice, the realization of robustness in navigation. These robots heralded a new wave in intelligent mobile robot research.

Table 1.1 shows a classification of the representative intelligent vision-based mobile robot research efforts, from the viewpoint of the navigation environment and the main research task.

As for the navigation environments, they are classified into two types: the *Arranged* ones and the *Unstructured* ones.

In an *arranged* environment, environmental conditions, such as landmarks or limits of obstacles, are set to be simple. Rather detailed knowledge about the structure of the navigation environment can be given (manually calibrated).

On the other hand, in an *unstructured* environment, environmental conditions are left as they are. Only an outline of the structure or even no information about the navigation environment is given in advance (we call this uncalibrated).

They are further classified into the *Indoor* navigation environments and the *Outdoor* navigation environments.

As for the research tasks, the *Attainment of quick reflexive response for real-time navigation*, the *Feasibility study of solutions to Artificial Intelligence (AI) problems* and the *Pursuit of navigation performance* are the main ones which correspond to the three directions of research shown in Fig.1.1, that is, the *Realization of reactive robust robot*, the *Acquisition of adaptability* and the *Pursuit of navigation performance.* The classification here is that which the author comprehends as the current (1996) realization stage, where most robot research addresses multiple research tasks and advances day by day.

Table 1.1: Classification of intelligent vision-based mobile robot research.

Research tasks	Navigation environment types			
	Arranged		Unstructured	
	Indoor	Outdoor	Indoor	Outdoor
Attainment of quick reflexive response for real-time navigation	Toy robots	Intelligent car (ETL) VaMoRs (BW Munich Univ.)	Insect robots (MIT)	HARUNOBU (Yamanashi Univ.)
Feasibility study of solutions to AI problems	Shakey, Flakey (SRI)	PVS (Fujitsu and Nissan)	HILARE (LAAS)	CMU Navlab (Carnegie Mellon Univ
Pursuit of navigation performance	Stanford cart (Stanford Univ.) Nuclear power plant robot (Toshiba et al.)	Alvin (Maryland Univ., Martin Marietta Denver Aerospace) Hyper Scooter (Tokyo Univ.)	CMU Rover (Carnegie Mellon Univ.) Yamabiko (Tsukuba Univ.)	Rocky 7 Mars Rover (JPL)

1.1.1 Attainment of quick reflexive response for real-time navigation

Indoor

Several toy robots equipped with infrared or ultrasonic ranging sensors which quickly avoid obstacles using simple algorithms are the most popular examples of indoor reactive robots, though they do not usually use vision sensors due to the high cost.

In 1986, R.A.Brooks of MIT proposed a novel type of reactive robots based on the **subsumption** decomposition[1]. Robots of this type had no clear missions, such as, reaching a final goal or making a map. Their main purpose was to maintain safety while moving around an office environment. In a sense, such robots were similar to insects.

His group has been developing several insect-type robots (*Genghis, Attila*) based on this concept[7].

Outdoor

A pioneering research effort for outdoor robots was the *intelligent car* at Japanese ETL which ran at speeds of 30 Km/h und used a vertically located stereo pair of cameras to detect obstacles by comparing vertical edges in both camera images[8]. Using special hardware for image processing, real-time navigation experiments were successfully done, but the vehicle's perception of the world was limited to a highly constrained environment.

In the *VaMoRs*[9] vehicle developed at BW Munich University, several processors detect the roadside edges in parallel, which makes it possible for the cart to run on a highway test course at a top speed of 100 Km/h. A real-time multiprocessor vision system was used for processing camera signals and for scene interpretation.

At Yamanashi University, the series of vehicles has been developed since 1983. HARUNOBU-4[10] has three wheels (two of them are driving wheels and other one is a free wheel). A TV camera platform which rotates horizontally is fixed on top of the robot with two infrared sensors. Several types of navigation behavior (Intersection-searching, Passage-segment-searching, Moving-along, Dead-reckoning-and-searching, Turning, and Avoiding-obstacle) have been tested on

the roads at the Yamanashi University campus[11].

1.1.2 Feasibility study of solutions to Artificial Intelligence (AI) problems

Indoor

Shakey, developed at the Stanford Research Institute at the end of the 1960's, was the result of pioneering research on intelligent mobile robots. Shakey was connected by wire with a huge mainframe computer, and moved shakily around a room, where several bricks were set, by processing the information from a rangefinder and a TV camera. The visibility graph (VGRAPH) description was used for modeling the environment. It sometimes took one day to avoid a simple static obstacle and plan a new path around it, due to the low computing power and the poor control mechanism.

In the mid-1980's, the SRI International AI Center developed its successor, *Flakey*, as a tool to explore several AI problems, such as evidential reasoning, planning using the procedural knowledge, rapid reasoning, motion analysis and natural language understanding. Flakey's hardware architecture consists of three layers. In the bottom layer, a power supply, a motor driving part, a bumper and twelve sonar sensors are implemented. A motor control processor and a main processor, which is a remodel of a SUN workstation, are in the middle layer, and a TV camera and a manipulator are in the top layer. Despite several attractive research projects, few papers reporting successful navigation of Flakey have been published.

The HILARE Project started in 1977 as an attempt at the LAAS (Laboratoire d'Automatique et d'Analyse des Systemes du CNRS) robotics group and other groups[12]. The *HILARE* was equipped with one free front wheel and two motorized rear drive wheels for locomotion. For perception, a video camera, an infrared triangulation system and a laser rangefinder were mounted on a two-axis scanning system and integrated to provide fast rough scene analysis consistent with navigation needs (for example, obstacle avoidance, decision-making, path-planning and search of minimum trajectory). The cell graph description for convex regions was used for hierarchical path planning.

Outdoor

The Personal Vehicle System (*PVS*) in Japan (Fujitsu Ltd. and Nissan Corp.) also successfully demonstrated experimental runs on an arranged test course (at 60 Km/h on straight sections, 15 Km/h on curves and 5 Km/h through intersections) using a high-speed image processor FIVIS/VIP for real-time white lane tracking[13].

At Carnegie Mellon University, an outdoor autonomous research project was started in 1984[14], with the aim of navigating through the campus sidewalk network using a small outdoor vehicle called the Terregator. In 1985, as part of the DARPA Autonomous Land Vehicle Project, a computer controlled van with onboard sensors and researchers named *Navlab*(Navigation Laboratory) was constructed[15]. For path planning, Navlab used the Annotated Map description, which consists of triggered positions, perceptual knowledge and control knowledge.

1.1.3 Pursuit of navigation performance

Indoor

The *Stanford Cart* and its successor, the *CMU Rover*, are the best-known examples of indoor mobile robot studies[16]. In the studies, the locations of critical points in images were detected by using the interest operator (to detect points which have high directional variance values), and the distance was measured by a stereopsis method using nine images from a slide-type stereo camera system. A description of an entire scene was constructed to find a location to where the robot could move. Then the robot moved to that location and repeated the same measurement to construct an environmental map. The uniform 3D-grid description was used for modeling the environment. The Stanford Cart took about 5 hours for a 20-meter move, with 20 % accuracy at best, lurching about 1 meter every 10 to 15 minutes before stopping again to take pictures, think and plan a new path. This was mainly due to a lack of computing power.

At Toshiba Corp., the author and co-workers developed a visual navigation system that allowed efficient movement in a nuclear power plant, as part of MITI's Advanced Robot Project[17]. The system con-

sisted of five subsystems: environment teaching subsystem, self-location measurement subsystem, obstacle detection subsystem, path planning subsystem and path following subsystem as shown in Fig.1.2[18]. Details of the obstacle detection algorithm used in the obstacle detection subsystem are given in Section 3. The necessary information for estimating self-location at critical points in the navigation environment is written in a MIL (Multi Information Local) map[19], which is interactively created in a remote-controlled run.

Fig.1.3 shows the robot developed by the project. Hitachi Ltd. developed the actuator part including four legs, Mitsubishi Heavy Industries Ltd. developed the manipulator part, Fujitsu Ltd. developed the vision sensor part, and Toshiba Corp. developed the system configuration and the navigation parts. A demonstration of real navigation in a static simulated nuclear power plant was successfully given in 1989.

At Tsukuba University, The series of small cart-type indoor robots *Yamabiko* have been developed for twenty years[20]. Basic functions for mobile robots, such as the landmark detection, the map generation and the path planning[21], based on the range information from sonar sensors (*Yamabiko* is a Japanese word whose mean is a echo) and a TV camera, have been extensively studied.

Outdoor

The Autonomous Land Vehicle (*ALV*) Project, which is a part of the DARPA Strategic Computing Program in the U.S.A., was intended to demonstrate the state-of-the-art in image understanding, artificial intelligence, advanced architectures and autonomous navigation.

In the ALV system[22] developed at Maryland University, the whole image was processed at the start(bootstrap stage) of the motion. After edge detection in the image, pairs of roadside edges were selected from the extracted edges, using the constraint of parallelism in the 3D space. Then, a local window was allocated to each of the pairs and processed. While the robot was moving(feed-forward stage), the whole image was no longer accessed. Based on the position and direction of the roadside edges in the previous image, the position of the next window to open was predicted, and the roadside edge was detected inside the window. By iterating this procedure, the direction of the robot progress was

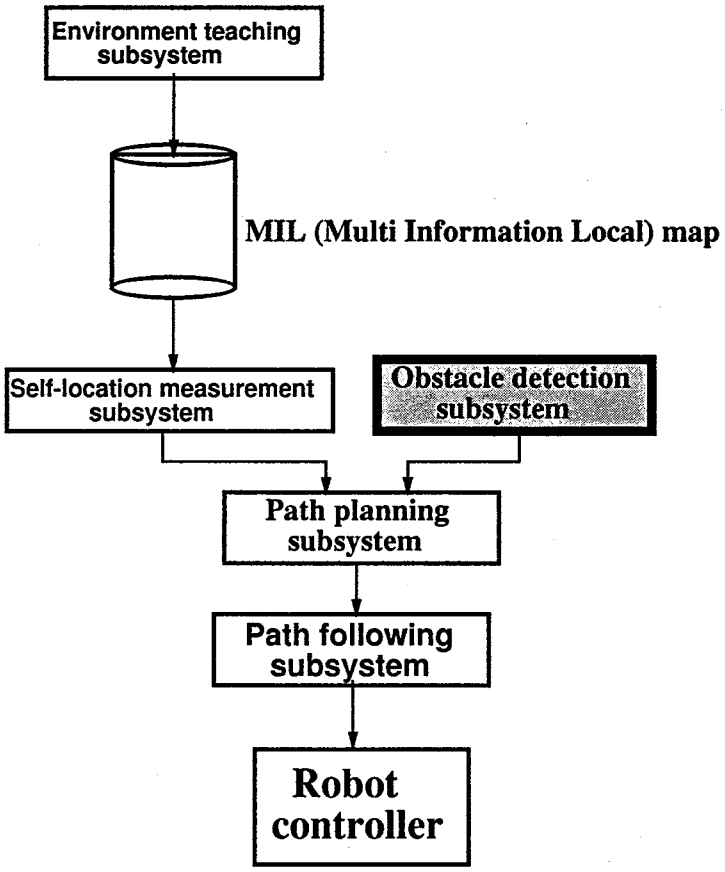


Figure 1.2: Configuration of visual navigation system for robot in nuclear power plants. Details of the obstacle detection algorithm used in the obstacle detection subsystem (hatched box) are given in Section 3.

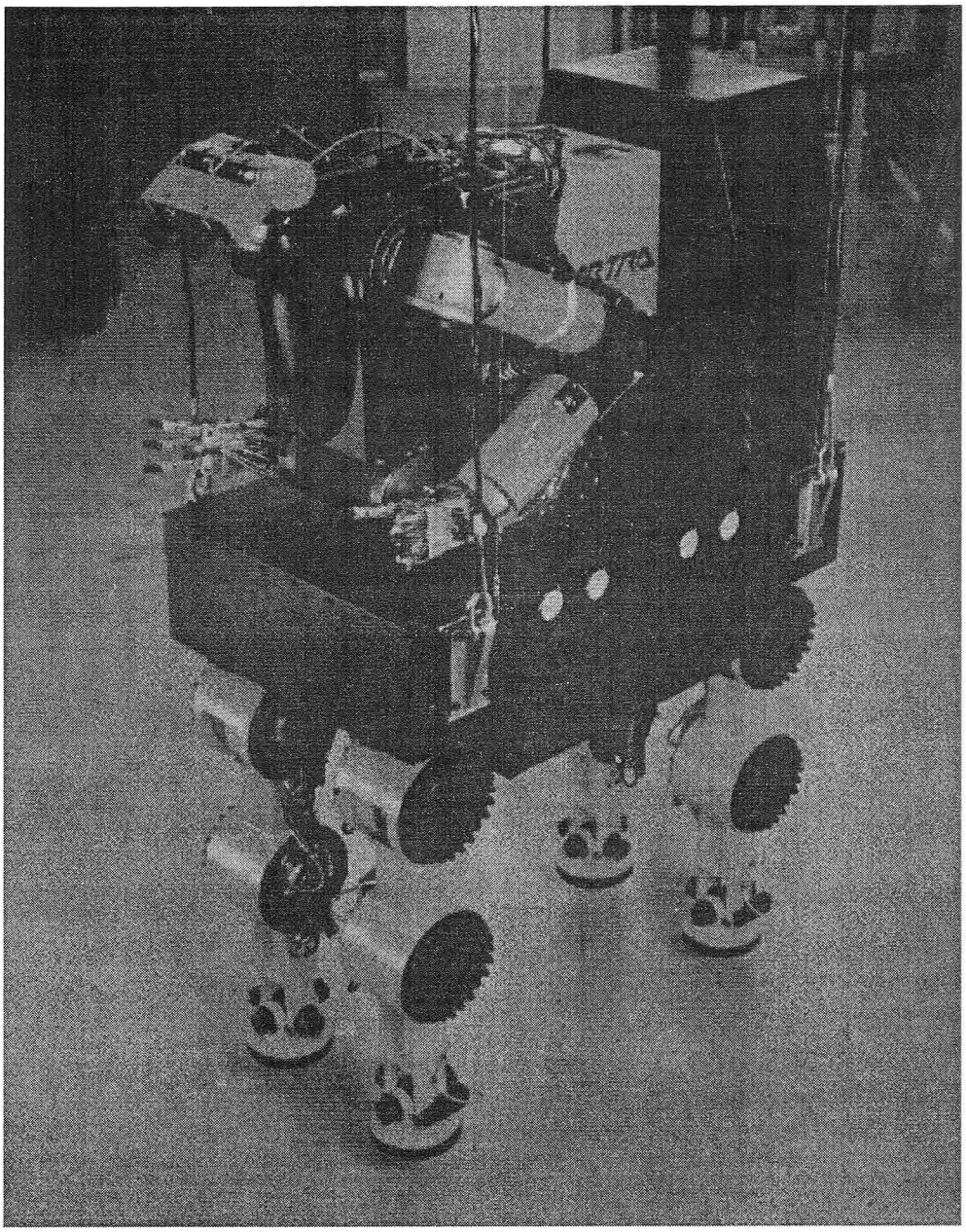


Figure 1.3: The robot developed as part of MITI's Advanced Robot Project.

controlled. A quad-tree description was used for three-level (long-range, intermediate-range and short-range) path planning.

Martin Marietta Denver Aerospace developed *Alvin*, equipped with an RGB video camera whose pan/tilt could be controlled, and a laser rangefinder[23]. The VITS (Vision Task Sequencer) system, whose prototype was originally developed at Maryland University, was extended to handle both video and range sensors for achieving road following capabilities. Alvin traveled 4.5 Km at speeds up to 10 km/h on a paved road.

At Tokyo University, a practical semi-autonomous mobile robot system called *Hyper Scooter*, on which a human could ride and share access to the environment through visual information, was developed[24]. A user could give instructions to the robot without skillful programming operations, and instead of human skill, a high speed tracking vision system [26] did the actual driving.

An example of a robot of this kind in an unstructured outdoor environments is the space robot prototype called the *Rocky 7 Mars Rover*, developed at Jet Propulsion Laboratory(JPL)[25].

Rocky 7 has a stereo camera system with a 5 cm baseline at both ends of the vehicle for avoiding static obstacles. An elevation map created by pyramid image processing is analyzed to detect abrupt changes in the height as obstacles.

1.1.4 Summary of issues in mobile robot research

In conventional mobile robot systems, generally, real-time navigation experiments have been successfully done only in simple arranged environments, such as a paved road, with long edge segments or painted with a single color that is easy to detect, where no moving obstacles appear. Autonomous navigation in real outdoor environments seems to be very difficult at present, except when human-guidance is used, such as in the case of Hyper Scooter, or in special static areas such as on Mars for Rocky 7.

On the other hand, reactive robots based on subsumption architecture have moved around successfully in real dynamic environments. However, clear navigation purposes, such as efficiently reaching a final goal, seem to be difficult to achieve.

1.2 The aim of this research

This thesis describes research and development concerning the architecture of a navigation controller and visual information processing to generate basic behaviors for autonomous mobile robots working in unstructured dynamic environments, such as offices, homes and streets; where,

1. Exclusive passage for robots is not established.
2. Independently moving objects such as humans exist.
3. Environmental conditions such as illumination or furniture positions frequently change.

In the classification of table 1.1, the robot we aim to realize is involved in the *Pursuit of navigation performance* group.

To achieve this aim, conventional systems from previous research are not applicable as they stand, because both the navigation environments and performance have been rather simple, compared with real situations.

1.2.1 Basic behaviors for mobile robots

In this thesis, an independent module which provides some behavior is called an **agent**. Each agent has a sensory input part, a processing of sensory information part and a driving command output part.

Here, a *behavior* means a directive action to realize a function needed for navigation.

Sophisticated vision-based agents are necessary when dealing with several obstacles in dynamic environments.

An autonomous robot system should posess the following three abilities.

- *Robustness:*

  In this thesis, this word represents the ability to cope with several unexpected situations (for example, the sudden appearance of obstacles), to assure **self-continuance**, though robustness in

Table 1.2: Examples of behaviors for mobile robots. Newly proposed behaviors in this thesis (Dead-end avoiding behavior and Obstacle recognition behavior) are enclosed in squares.

Important ability	Basic behavior
Robustness	Obstacle detection Free-space detection Collision avoidance
Efficiency	Target tracking Path following Dead-end avoiding
Intelligence	Path planning Map generation Obstacle recognition

general is necessary for all behaviors. This includes *adaptability* to flexibly accommodate environmental changes.

- *Efficiency:*

  The ability to achieve its mission (for example, safe navigation to a final goal), by the best cost-performance.

- *Intelligence:*

  The ability to evolve itself to attain better performance (for example, environment-map making for a subsequent navigation).

Table 1.2 shows the typical examples of the behaviors of mobile robots.

For example, obstacle detection, free-space detection and collision avoidance belong to the Robustness ability. Though they have a layered relationship at the functional level, that is, the collision avoidance function consists of the obstacle detection function and the free-space detection function, in this table they are listed independently because

the original behaviors are independent. A quick response time is necessary for obstacle detection, while accuracy is required for collision avoidance.

A prediction process can improve both the reliability and the response time for obstacle detection. A static obstacle detection technique and a moving obstacle detection technique are explained in Section 3 and Section 4, respectively.

Target tracking, path following behavior and dead-end avoiding belong to the Efficiency ability. In several studies, these behaviors have been functionally developed to ensure a quick response time (for example, white lane tracking) in conventional reactive mobile robot systems.

Path planning, map making and obstacle recognition belong to the Intelligence ability. Flexibility is the most important factor for these behaviors.

Because every function based on a behavior has a possibility of failure, it is necessary to implement countermeasures against these failures. Dead-end avoiding is an example of a special behavior for recovering from dead-end situations.

Newly proposed behaviors in this thesis are enclosed in squares in table 1.2, that is, **the dead-end avoiding behavior** and **the obstacle recognition behavior**.

Obstacle recognition is useful to achieve an intelligent collision avoidance function according to the type of obstacle.

The design of these behaviors is given in Section 2 and Section 4, respectively.

1.2.2 Desired control systems for mobile robots

The behavior-based composition [27] is recognized as the most suitable control system for robots working in a dynamic environment.

The main issues of behavior-based systems are as follows:

- *Situatedness:* Instead of acquiring a symbolic model description of an outer world to manage abstract problems, the outer real world itself should be managed. (The world is its own best model.)

- *Embodiment:* Intelligence cannot be realized without a body, as

intelligence without a body does not exist. That is, robot research should advance by developing real working robots.

- *Intelligence:* Intelligent behavior emerges from interaction with the real world.
- *Emergence:* Intelligence is a basic element which emerges from interactions among several elements of a robot system, instead of as an independent local one.

In summary, intelligence emerges from the interaction of several basic behaviors and the outer real world. For this reason, any problem for intelligent mobile robots, such as navigation or learning, should be divided into the basic behaviors (agents).

Each agent can be realized by rather simple procedures because it has a unique purpose and complex world models are not necessary in many cases. For example, an obstacle detection agent does not have to discriminate types of obstacles or find a path to avoid. It just detects objects which may interfere with the robot itself. No description of the outer world is necessary. Because of the simplicity of these agents, they can be realized robustly, and a robot control system, composed of these agents, is highly adaptive to changes in environmental conditions.

The most popular example of behavior-based decomposition is the subsumption[1]. In subsumption architecture, basic behaviors are layered in order such as, *avoid objects, wander, explore, build maps, monitor changes, identify objects, plan changes to a world* and *reason about behavior of objects.* Robustness is secured by the distribution and parallel activation of these behaviors.

The major problem of subsumption systems is the trade-off between robustness and efficiency (and intelligence) in the same control framework.

A new control architecture to attain both robustness and efficiency is proposed in Section 2.

Chapter 2

Control architecture

The author is motivated by a desire to bridge the gap between reflexive/reactive behaviors for self-continuance and purposive navigation behaviors in order to navigate a single mobile robot in a dynamic environment to a final goal robustly and effectively, while preserving the merits of behavior-based architecture.

In spite of several improvements in recent behavior-based researches, conventional behavior-based systems still seem to have a weakness when purposive navigation behaviors are combined with reactive/reflexive behaviors for self-continuance, especially from the viewpoint of failure recovery.

To overcome this difficulty, special agents to recover from several failures are newly developed, and all agents are classified into three groups according to the tasks, that is, to keep robustness for self-continuance (reflexive), to attain the mission of the robot (purposive) and to recover from failures (adaptive).

These grouped agents are directly connected to a navigation actuator controller named Motion Executor, which determines the motion (velocity and steering) command of the robot.

An extended energy-minimizing method is developed to produce an appropriate velocity and steering commands for the robot navigation.

To prove the effectiveness of the architecture, simulation experiments and real navigation experiments using a compact cart-type robot **BIRDIE** have been done.

2.1 Related work

As for control architectures, two types have been proposed to date: *functional decomposition* and *behavior-based decomposition.*

Robots with functional decomposition normally use an *SMPA* (Sence-Model-Plan-Act) framework as shown in Fig.2.1. At first, an external world is observed using vision sensors such as CCD cameras or ultrasonic ranging sensors and a description of the observation is generated. Next, an internal path-planner searches for the optimal route by matching the current description with a prepared model of the environment. The robot then moves following the planned path.

This SMPA architecture is suitable for efficient robot navigation. On the other hand, this architecture has weakness in sensing and processing from the viewpoint of robustness against several kinds of failures which may occur frequently in dynamic and complex environments. Once an error occurs in a process, the prepared motion plan can no longer be executed.

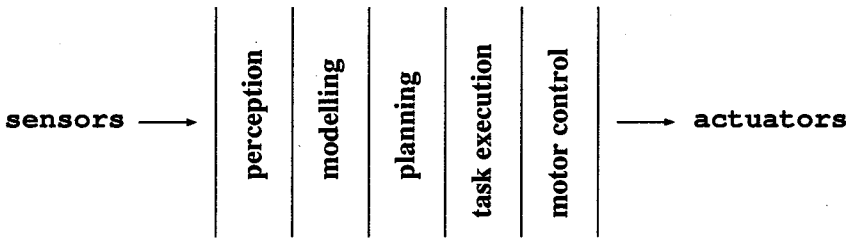


Figure 2.1: SMPA framework. The processing of the SMPA framework is sequential, such as, Sensing - Modeling - Planning - Action.

Behavior-based decomposition was first proposed as a *subsumption architecture* by Brooks[1] as a means to overcome this difficulty. In the subsumption architecture, basic behaviors are layered in the order such as, *avoid objects, wander, explore, build maps, monitor changes, identify objects, plan changes to a world* and *reason about behavior of objects*, as shown in Fig.2.2. Robustness is secured by the distribution and parallel activation of these behaviors.

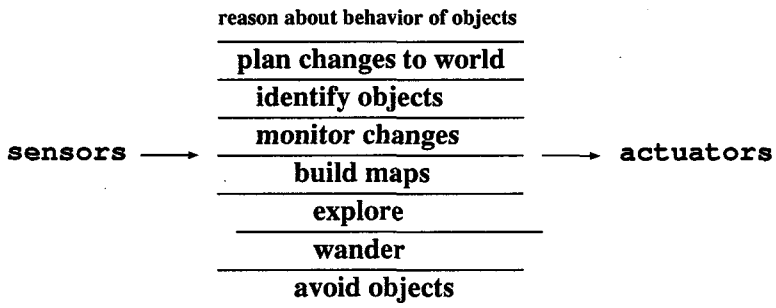


Figure 2.2: Subsumption framework. The processing of the subsumption framework is parallel and layered in the order of robustness.

Recently, several advanced behavior-based systems have been proposed to give a purposive capability to a robust reflexive/reactive configuration for self-continuance.

Arkin proposed a motor schema-based architecture[28]. A schema is a methodology used to describe an interaction between perception and action. Each individual motor schema corresponds to a reflexive behavior. A dynamic network was utilized instead of a layered configuration.

Maes proposed a method of selecting an action for an autonomous agent by modeling as an emergent property of an activation/inhibition dynamics among the actions and the environment[29].

Noreils presented a three-level (functional, control system and planning) configuration for indoor mobile robots[30]. The functional level provides the basic capabilities (behaviors) of the robot, such as, *wall-following, obstacle-avoidance, tracking, sonar-orientation* and *vision-position.* The control system level manages the robot's modules in order to obtain the behavior corresponding to the current mission. The planning level consists of the *general planner* and the *path-planner.* It uses an AND/OR goal tree from a given goal into subgoals. Each leaf of the tree sends a command to the execution control. For navigation, the intermediate goal EXECUTE-PATH is created and the general planner sends a request to the path-planner to compute a path between two

locations. With the returned path, the general-planner creates a mission such as MOVE-NEAR or MOVE-TO. This configuration is more flexible than conventional subsumption because the functional level can be programmed by the control level in accordance with the mission of the robot. The planning part comprises two rule-based modules: a general-planner and a path-planner which prevents some failures from occurring in navigation by applying prepared rules.

Hartley developed a prototype of a behavior-based airplane controller[31]. In the implementation, three levels of behaviors (top, middle and low) were used. For example, *control approach velocity behavior* belongs to the low level. *Turn behavior, fly along a line behavior* and *fly to a point behavior* belong to the middle level. Each behavior consists of different low level behaviors. For example, *final approach behavior* starts with *landing gear behavior* in low level, then uses *stay behavior* at the *correct vertical behavior.* To save the *lack of modularity* problem, control(Inhibit, Suppress) connections among behaviors were utilized. Other possible methods for the arbitration of behavior were also suggested in this paper including priority lists and hysteresis.

Despite several improvements in recent behavior-based systems, these systems still seem to have a weakness when purposive navigation behaviors are combined with reactive/reflexive behaviors for self-continuance, especially from the viewpoint of failure recovery. Planning-only failure recovery and behavior arbitration by restricted priority list may negate the essential merits of behavior-based architecture, that is, robustness and flexibility. Interactions among agents may occur and robustness against failures of some agents, which is the main target of behavior-based architecture, is not necessarily guaranteed. In addition, a failure recovery mechanisms against undetectable individual failures and total failures, such as infinite-loop-motion, have not been examined in conventional systems.

2.2 System design

A control architecture, which is suitable for navigating a single mobile robot in an office to a final goal robustly and effectively, is proposed. We assumed that the route to the goal is roughly given manually as a

sequence of subgoals in advance, and an arrival to a subgoal or the goal is determined by the robot itself.

In the case of navigation in unstructured dynamic environments, such as offices, stations and streets, several failures (for example, missing of subgoals or a final goal) may occur frequently. To cope with these situations, we examined failure types to be coped with first, then prepared special adaptive agents which provide behavior output to recover from these failure situations using heuristics, and finally installed them in the system.

Those failures are classified into two types: *the individual failure* which occurs in the process of generating a behavior from sensing results and *the total failure* which occurs in a situation when dominant behavior does not coincide with the achievement of the global mission of the robot, that is, it greatly reduces efficiency of robot navigation.

Table 2.1 shows the failures to be coped with by the system.

Table 2.1: Type of failure

	Individual Failure		**Total Failure**
	Detectable	Undetectable	
Self-Continuance Level		Missing moving obstacle in Obstacle-Avoider	Infinite-loop stopping
Purposive Navigation Level	Missing target in Target-Searcher Missing target in Target-Tracker	Detecting false target in Target-Searcher Mis-tracking of target in Target-Tracker	Infinite-loop motion sequence

The individual failures are still classified into *the detectable failures* in the agent itself and *the undetectable failures.* For example, the missing target failure of target-searching and the missing target failure of

target-tracking are detectable failures in the agents themselves. On the other hand, the detecting false target failure of target-searching, the miss-tracking of target failure of target-tracking and the missing obstacle failure of a reflexive behavior agent are undetectable failures.

The total failures generally appear in infinite-loop-motion situations. For example, the infinite-loop-stopping failure when a robot touches an obstacle and a safety circuit shuts off a power source and the infinite-loop-motion-sequence failure when a robot becomes sandwiched between multiple close obstacles are total failures.

We examined only these two total failures in the system and did not consider more complex failures, such as interference among multiple robots, as we considered a navigation of one robot in an office environment.

We established principles to cope with the above failures.

As for detectable individual failures, the system immediately recovers from the failures by using alternative agents, which provide behavior output for recovery, and prevents the loss of navigation efficiency and robustness. Multiple independent obstacle detection agents using various kinds of sensors are prepared and the priority-based behavior arbitration method is adopted for the reflexive agents, because robustness has to be secured for self-continuance even for undetectable individual failures. In addition, as for total failures, special adaptive agents are prepared to escape from the infinite-loop-motion situations so as to avoid the great loss of navigation efficiency.

To realize these principles in the control system, the author, Onoguchi, Kweon and Kuno proposed a new behavior-based architecture [32] in which all navigation behaviors were clustered into three task-oriented groups instead of a single layered configuration.

2.2.1 Constitution of group-based multi-agent system

The main problem in designing a navigation control system is *how to reach a compromise between the navigation efficiency and robustness against several failures* mentioned in the last section. If no failure happens during navigation, the best system is the configuration of the

minimal reflexive agents for self-continuance and purposive navigation agents, in which the driving command is determined by directly switching the behavior outputs of these agents.

However, when the number of agents increases by adding obstacle detection agents to improve robustness or adding adaptive agents to recover from the failures, the merit of the original behavior-based architecture is lost due to the complexity of behavior arbitration in multiple agents. In that case, preparation of a few groups which involve agents with similar tasks and behavior arbitration among these groups are considered to be superior from the viewpoint of navigation efficiency. In addition, the group-based system has the merit that the alteration of the whole system is not necessary when new agents are added to improve the ability of failure recovery.

In the proposed system, all navigation behaviors are clustered into three task-oriented groups, that is, a **Reflexive Behavior Agent Group**, a **Purposive Navigation Behavior Agent Group** and an **Adaptive Behavior Agent Group**.

The number of groups is determined in accordance with the mission of navigation. The number of our system is three, because we assumed an indoor navigation with given subgoals and a final goal.

While the first two groups exist in conventional systems, the third Adaptive Behavior Agent Group is newly proposed by the author[33].

Fig.2.3 shows the configuration of the system.

2.2.2 Driving command determination method

As for the method of determining a driving command in the system, the conventional SMPA architecture has a problem because the mechanism is complex and easily fails when some parts of intermediate processes fail or are interrupted. Though the subsumption architecture is simple and robust against a failure of an intermediate process, the fixed priority-based driving command determination by selecting behaviors of activated agents (arbitration) tends to select lower-level behaviors which are not good to achieve efficient navigation.

Though conventional systems have unique mechanisms, two kinds of arbitration mechanisms are prepared in our system: the arbitration mechanism *inside each group* and that *among different groups.*

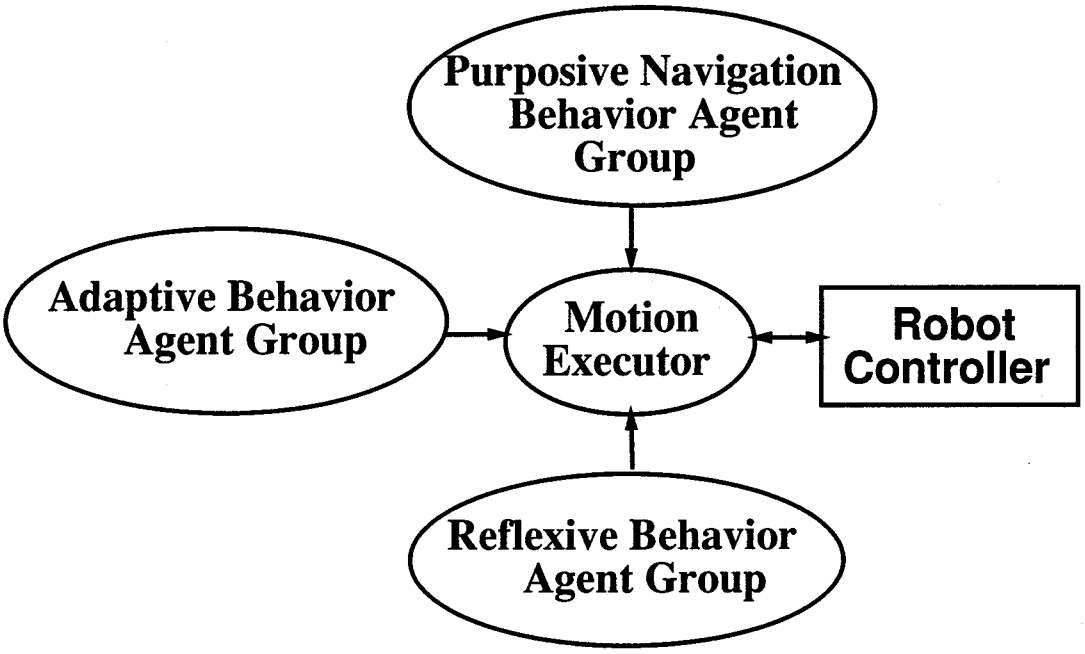


Figure 2.3: System configuration(group level). Three task-oriented groups (the Reflexive Behavior Agent Group, the Purposive Navigation Behavior Agent Group and the Adaptive Behavior Agent Group) are directly connected to the Motion Executor which determines a driving command and sends it to the Robot Controller.

The agent arbitrators independently prepared for each group determine appropriate commands first, and then the driving command of the robot is determined uniquely by selection of these commands from every group based on priority of the groups. Both robustness against failures and efficiency in navigation are realized by this distributed arbitration mechanism.

Each agent always calculates a vector from a current position to a position where the robot should move next, from sensing results and history of selected driving commands, as behavior output. The methods of calculating driving command vectors are described in the next section. When the length of the vector is larger than the threshold value, the agent is judged to be active. This judgment information is utilized to select behaviors of agents.

A Reflexive Behavior agent for self-continuance is designed to provide an output vector only when it detects obstacles. The agent is judged active when it provides an output, including 0 (STOP command) vector provided by the static obstacle detection agent.

A Purposive Navigation Behavior agent calculates the position of targets corresponding to current subgoal or goal, and detects the failure of target detection in itself. When the agent detects the failure, it suddenly provides 0 (STOP command) vector and is judged to be inactive.

An Adaptive Behavior agent always provides a driving command vector while sensors are active. The judgment of activity in this case is done by comparing the vector length with a threshold value.

A driving command for motion actuators is determined by simply selecting an appropriate behavior from the arbitrated results in three groups. The selection, that is, the arbitration among different groups, is done by the following rule:

First of all, a behavior from the Reflexive Behavior Agent Group for self-continuance is preferred to other groups except for infinite-loop-motion situations because safety is the most important consideration.

Secondly, a behavior from the Purposive Navigation Behavior Agent Group is preferred to achieve efficient navigation as long as the safety is guaranteed.

When a failure of the Purposive Navigation Behavior Agent Group or an infinite-loop-motion situation occurs, a behavior from the Adap-

tive Behavior Agent Group is selected to recover from the failure.

The failure detection method is shown in the next section.

The arbitration methods in the three groups are as follows:

As for the Reflexive Behavior Agent Group, the arbitration priority is determined according to the reliability of sensor observation of each agent, and selection is done based on this priority. That is, the active agent with the highest priority is selected. In a neighborhood of a subgoal or a final goal, a target object which features the subgoal or the goal position is supposed to exist. The agents which search or track these targets are installed in the Purposive Navigation Behavior Agent Group.

As for the Purposive Navigation Behavior Agent Group, the priority is determined by the sequence of robot navigation. That is, target searching and tracking agents are selected in order, following the sequence of subgoals and a final goal in the route given manually, and navigate the robot to the final goal efficiently. Instead of maintaining a detailed geometric map, only a topological information (the sequence of targets) is necessary.

As for the Adaptive Behavior Agent Group, the Motion Executor selects an appropriate agent which recovers from failures, such as missing of targets and a infinite-loop-motion situations, according to the type of failures.

Fig.2.4 shows the detailed configuration of the system.

2.3 Detailed explanation of agents

2.3.1 Energy-based motion determination algorithm

Acoustics sensors such as ultrasonic ranging sensors provide an inexpensive means to obtain range information around the robot by computing the echo travel time.

Kweon, the author, Kuno and Onoguchi have presented the energy-based motion decision algorithm using multiple range information around a robot[34], [35].

In a behavior-based mobile robot, each navigation behavior agent

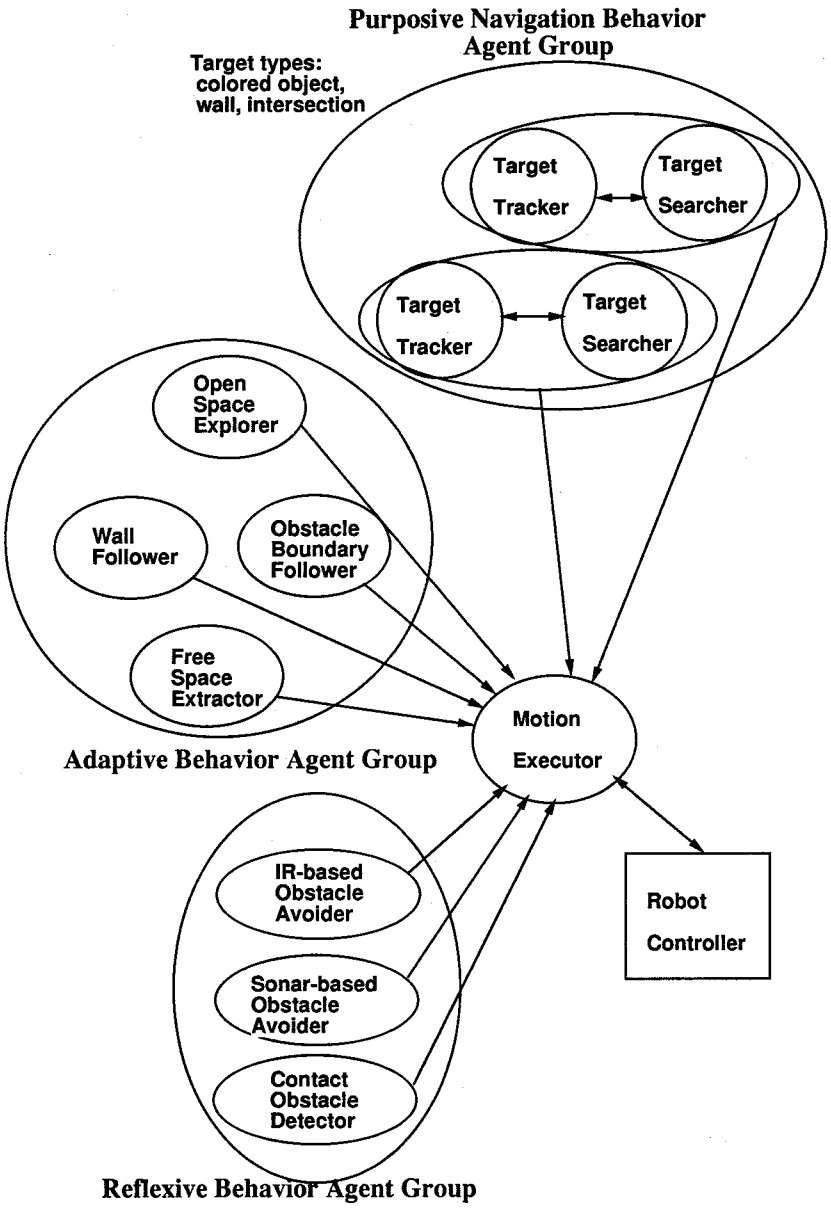


Figure 2.4: System configuration(agent level). This figure shows all agents involved in Fig.2.3.

computes a driving command directly from sensor data without building internal representation. Therefore, behaviors become more robust when multiple sensor data are used to compute driving commands.

Here a method to compute a position and a direction of a robot are presented by fusing multiple sensor data based on active contour models.

To determine the best position and orientation of the robot using the range readings from multiple sensors, the energy-minimizing contour model, known as the **snakes**[36] is utilized.

In the *snakes* algorithm, a solution can be seen as realizing the equilibrium of the forces acting on the contour model. We apply the idea to compute the position and orientation of the robot, at which the equilibrium of all forces is maintained.

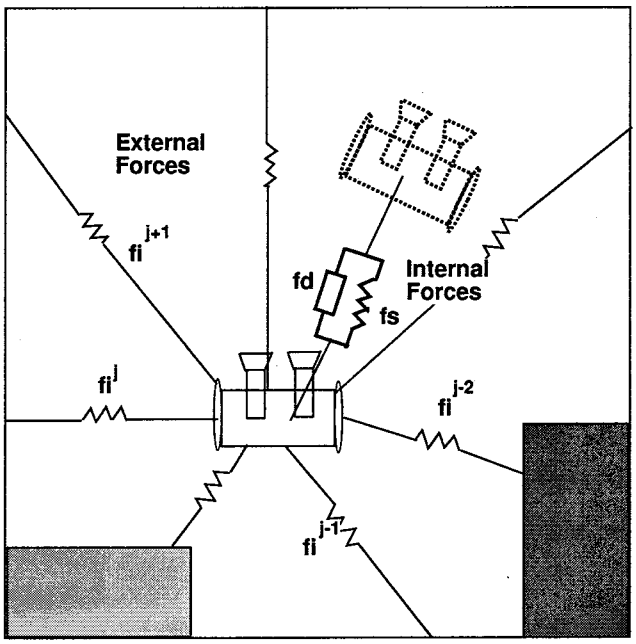


Figure 2.5: The energy-based motion decision. The dashed robot shows the previous position. Internal forces, $\mathbf{f_s}$ and $\mathbf{f_d}$, try to keep the robot's trajectory smooth.

1. As internal forces, we consider a smooth force, $\mathbf{f_s}$, from the smoothness constraints on the robot's trajectory and a damping force, $\mathbf{f_d}$, from the robot dynamics.

2. As external forces, the range forces, $\mathbf{f_i}$, from multiple range measurements, push the robot to the location where equilibrium is achieved between the range forces (the safest position), and other external forces, such as the target forces, can be added to attract the robot.

Forces acting on the robot include internal and external forces as shown in Fig.2.5[37]:

Here, a robot is bound to 2D (X-Y) space which is described by 2D Cartesian coordinates. Fig.2.6 shows the definition of symbols used in the following explanation.

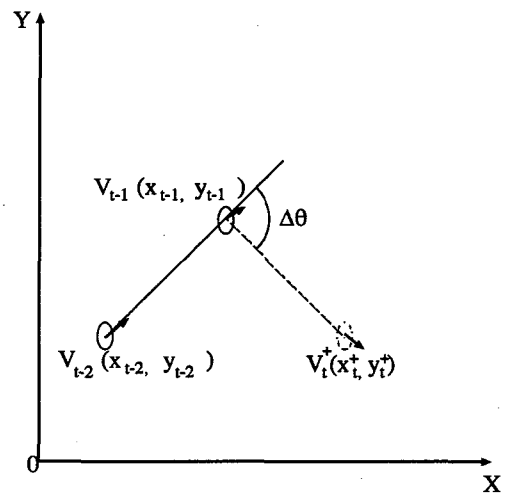


Figure 2.6: The definition of symbols and 2D Cartesian coordinates. **0** means an original position of the 2D coordinates. A location of the robot is shown as a central point of a small ellipse. A small arrow shows a direction in which the robot starts to move from each location. $\mathbf{v_t^+}$ is the *t-th* location computed from the smoothness constraints.

The location of the robot is presented by a 2D vector

$$\mathbf{v_t} = (x_t, y_t) \tag{2.1}$$

where (x_t, y_t) is the Cartesian coordinate of the robot's position on the floor at time t. Then, the internal smoothness force is computed by

$$\mathbf{f_s}(\mathbf{v}) = \alpha(\mathbf{v_t^+} - \mathbf{v_t}) \tag{2.2}$$

where $\mathbf{v_t^+}$ is the *t-th* location computed from the smoothness constraints, and $\mathbf{v_t}$ is the current robot position. α is a parameter to control the smoothness in the robot movement. Based on simple trigonometry, it is easy to compute the robot's location, $\mathbf{v_t^+}$, satisfying the smoothness constraint.

$$\mathbf{v_t^+} = (x_t^+, y_t^+) \tag{2.3}$$

$$x_t^+ = 2x_{t-1} - x_{t-2} - 2(y_{t-1} - y_{t-2})\tan(\frac{\Delta\theta}{2}) \tag{2.4}$$

$$y_t^+ = 2y_{t-1} - y_{t-2} - 2(x_{t-1} - x_{t-2})\tan(\frac{\Delta\theta}{2}) \tag{2.5}$$

where $\Delta\theta$ is the angle between the two lines formed by three robot locations as shown in Fig.2.6.

$$\mathbf{f_d}(\mathbf{v}) = d(\mathbf{v_t} - \mathbf{v_{t-1}}) \tag{2.6}$$

where $\mathbf{v_t}$ and $\mathbf{v_{t-1}}$ are the *t-th*(current) and *(t-1)-th*(previous) robot locations, and *itd* is a control parameter to effect the dynamics of the robot. The range force for each sensor is computed as

$$\mathbf{f_i^j}(\mathbf{s}, \mathbf{v}) = k(\mathbf{s_j} - \mathbf{v_t}) \tag{2.7}$$

where $\mathbf{s_j}$ represents the Cartesian coordinate values for the *j-th* range reading, and k is the range force parameter which controls the effect of the range force.

From Eqs.(2.2),(2.6),(2.7), a new location of the robot, satisfying the force equilibrium condition from multiple sensors, is simply computed by

$$\mathbf{v_{t+1}} = \mathbf{v_t} + \sum_{j=0}^{n} \mathbf{f_i^j}(\mathbf{s}, \mathbf{v}) + \mathbf{f_d}(\mathbf{v}) + \mathbf{f_s}(\mathbf{v}) \tag{2.8}$$

where n is the number of sensors. The new location is controlled by three parameters: α, d and k for the smoothness constraint, the damping force and the range force, respectively.

2.3.2 Reflexive Behavior Agent Group

The roles of the agents belonging to the Reflexive Behavior Agent Group are to maintain the safety of humans and the robot itself, that is, obstacle detection recognition and collision avoidance. We adopt a multi-sensory configuration to avoid the danger caused by a failure of obstacle detection.

In the system, the Contact Obstacle Detector using rubber sensors, Sonar-based Obstacle Avoider using ultrasonic ranging sensors and IR(infrared)-based Obstacle Detector, especially prepared for human obstacles, are installed. Here a motion determination algorithm of the Sonar-based Obstacle Avoider is explained because these reflexive agents use similar ones.

Sonar-based Obstacle-Avoider

The Sonar-based Obstacle-Avoider(**O-A**) only uses range measurements reflected from nearby obstacles, because any range measurements that are greater than a specified distance guarantee the robot's safety. For the O-A, the range forces from nearby obstacles act as repulsive forces on the robot, while in the standard energy-based motion decision, the range forces act as attractive forces. The repulsive range forces pushe the robot to a new location.

The range force is inversely proportional to the range measurement. In other words, nearer obstacles provide greater repulsive forces.

In (2.7), the range force $\mathbf{f_i^j}(\mathbf{s}, \mathbf{v})$ is replaced by a repulsive range force:

$$\mathbf{f_i^j}(\mathbf{s}, \mathbf{v}) = \begin{cases} -k\mathbf{s_j} & \text{if} \quad \rho_j \leq \rho_s \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where $\mathbf{s_j}$ is a Cartesian coordinate value of the *j-th* sonar measurement ρ_j, and ρ_s is the minimum safety distance for the robot.

2.3.3 Purposive Navigation Behavior Agent Group

The role of the agents belonging to the Purposive Navigation Behavior Agent Group is to assist in navigating the robot to the final goal efficiently. The navigation sequence to reach the goal is realized as the specified agent pairs consisting of target-searching and target-tracking. Each target corresponds to an object which characterizes a subgoal and a final goal to be reached.

We have developed these three types of purposive navigation behavior agents by using features of targets in camera images.

Color-Tracker

This agent simply detects a region whose hue value is in a range of target hue values in input color images, and calculates the direction of the target.

When the focal length of the camera is f, and the distance between the optical center and the center of extracted region is d, The direction angle θ_c is calculated as follows[34]:

$$\theta_c = \arctan \frac{d}{f} \tag{2.10}$$

Wall-Tracker

Part of an indoor environment can usually be described by sets of parallel 3D lines, such as walls and corridors. In an image, they intersect at a vanishing point.

Using this fact, this agent extracts a vanishing point (x_v, y_v), and calculates an angle needed to align a robot with the wall-steering angle θ_w as follows:

$$(x_v, y_v) = (f\frac{\tan\alpha}{\cos\delta}, f\tan\delta) \tag{2.11}$$

$$\theta_w = \arctan \frac{y_v}{f} \tag{2.12}$$

where f is the focal length of the camera, and δ and α indicate pan and tilt angle of the camera with respect to the world coordinate.

Intersection-Tracker

This agents detect an intersection using a simple heuristic: *intersections exist at the end of walls.*

The algorithm works in the following steps[34]:

1. Compute a vanishing point.
2. Find two corresponding straight lines on the floor, which form the vanishing point.
3. Extract vertical lines on each straight line, located at approximately the same distance from the robot.

The steering angle of the robot is determined by,

$$\tan \theta_i = \frac{C_{avg} - C_i}{f} \tag{2.13}$$

where f indicates the focal length, C_i is the column value of the image center, and C_{avg} is the average column value of two vertical edges.

2.3.4 Adaptive Behavior Agent Group

The role of the agents belonging to the Adaptive Behavior Agent Group is to bridge the gap between the reflexive behavior for self-continuance and the purposive navigation behavior, that is, to provide behaviors appropriate for recovering from the undetectable failures in the purposive navigation behavior agents and the total failures in infinite-loop-motion situations.

In the system, we have prepared these three agents using sonar sensors.

Free-Space-Explorer

A Free-Space-Explorer(**F-S-E**) generates a driving command to push a robot to the largest open space, which corresponds to the safest area for

the robot. To achieve this goal effectively, the F-S-E uses the energy-based method, with a range force parameter, k, at around unity.

$$\mathbf{f_i^j}(\mathbf{s}, \mathbf{v}) = k\mathbf{s_j} \tag{2.14}$$

where $\mathbf{s_j}$ is a Cartesian coordinate value of the *j-th* sonar reading. This results in the range forces from sensors having a major influence and the robot internal energy having a small effect. In this case, the robot moves to the location where equilibrium between range forces is achieved.

Obstacle-Boundary-Follower

The Obstacle-Boundary-Follower(**O-B-F**) uses an algorithm similar to the O-A. In the case of the F-S-E and O-A, the range forces from nearby obstacles are either attractive or repulsive. For O-B-F, range measurements from nearby obstacles are used to compute tangential forces.

In (2.7), the range force $\mathbf{f_i^j}(\mathbf{s}, \mathbf{v})$ is replaced by a tangential range force:

$$\mathbf{f_i^j}(\mathbf{s}, \mathbf{v}) = \begin{cases} k(\mathbf{s_j} - \mathbf{s_{j-1}}) & \text{if} \quad \rho_j \leq \rho_{th} \text{ and } \rho_{j-1} \leq \rho_{th} \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

where $\mathbf{s_j}$ is a Cartesian coordinate value of the *j-th* sonar measurement ρ_j, and ρ_{th} is a threshold distance for nearby obstacles. The robot is only pushed by these tangential forces. As long as good range measurements are available without any systematic error, such as specular reflection of sonar, the robot will never collide with obstacles, because it always moves along the tangential directions of nearby obstacles.

The failure of this behavior can be easily detected by the null driving command from this behavior which corresponds to these two possible situations:

- There are no obstacles to follow.
- Obstacles are not detected due to sensor errors.

Additional sensing from the deliberate motion of the robot or other sensor is required to detect the real failure. Instead of that, we used the exploratory rotational motion of robot to obtain multiple range measurements at one robot location. These multiple range measurements are then used to confirm the failure of the behavior.

Wall-Follower

The Wall-Follower(**W-F**) uses the same algorithm as the Obstacle-Boundary-Follower(**O-B-F**). The only difference is that the object to follow is not an unknown obstacle but a known wall existing in an environment.

Open-Space-Explorer

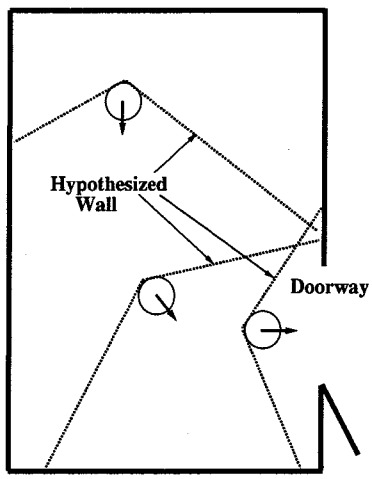


Figure 2.7: The Open-Space-Explorer using the Free-Space-Explorer. A circle shows a position, and an arrow shows the heading direction of the robot. Dashed lines behind a circle show the hypothesized wall which pushes the robot to free space.

In indoor navigation, a robot often needs the function of exploring open space to find an exit in a room by just using robust reflexive

behaviors for self-continuance, without using any sophisticated vision systems or algorithms. We developed the Open-Space-Explorer(**O-S-E**) based on the behavior of the F-S-E.

$$\mathbf{f_i^j}(\mathbf{s}, \mathbf{v}) = k\mathbf{s_j} \tag{2.16}$$

where a range force parameter k is set at around unity.

The basic idea is to hypothesize an oblique wall immediately behind the robot, as shown in Fig.2.7, and to update the actual sonar measurements with the synthetic range readings reflected from this hypothesized wall. Then, with these updated range readings, a new robot location is computed by the algorithm of the F-S-E.

2.4 Motion Executor

Fig.2.8 show the mechanism of determining driving commands schematically. The driving command for actuators is determined in the Motion Executor by simply selecting an appropriate behavior from the arbitrated results in three groups following the rule mentioned in the previous section.

The agent arbitration mechanism inside each group is prepared inside the subsystem (**the Motion Executor**) which controls the actuators of the robot as mentioned in the last section.

The Reflexive Arbitrator for Reflexive Behavior Agent Group adopts the fixed priority-based arbitration. The priority is determined according to the reliability of sensor observation.

The Purposive Arbitrator for Purposive Navigation Behavior Agent Group adopts the mission-based arbitration. The priority is determined by the sequence of robot navigation.

On the other hand, no formal arbitration is necessary for the Adaptive Behavior Agent Group because each agent always provides a behavior to stably approach free space. The Motion Executor can select any active behavior when it requires it.

In addition, a rule-based arbitration can be installed according to the type of failures from the viewpoint of increasing the navigation efficiency.

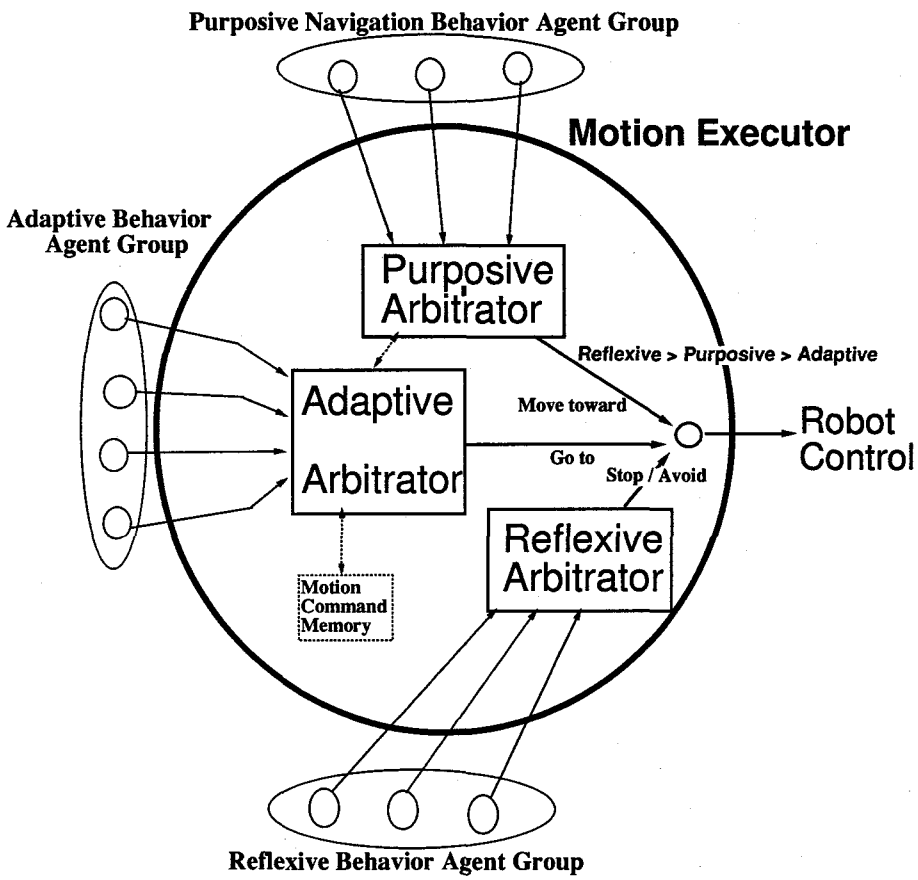


Figure 2.8: Mechanism of driving command determination. In the Motion Executor, two kinds of arbitration systems are prepared: the arbitration system *among different groups* and that *inside each group.*

The possible rules are these two kinds: *Missing target failure recovery rules* that indicate an agent which recovers from an agent's failure in Purposive Navigation Behavior Agent Group, and *Infinite-loop-motion failure recovery rules* that indicate an agent which recovers from infinite-loop-motion situations.

Suppose that A is a purposive navigation agent which fails in target tracking or searching. $Agent(i)$ is an adaptive agent which recovers from the failure, and n is the number of possible agents to select. V_A and $V(i)$ are driving command strength of the agent A and $Agent(i)$, respectively, and calculated as the length of the vector from current position to the next position where a robot should move.

Then, the Missing target failure recovery rule is described as,

if $V_A = 0$
  if $V(1) > Th(1)$ Select Agent(1)
    else if $V(2) > Th(2)$ Select Agent(2)
      .....
        else if $V(i) > Th(i)$ Select Agent(i)
          .....
            else if $V(n-1) > Th(n-1)$ Select Agent(n-1)
              else Select Agent(n)

Here, $V = 0$ means that the failure of the agent A is detected by agent A itself and no motion command is provided to the Motion Executor. $Th(1)$ and $Th(2)$ are threshold values to judge whether the agents, $Agent(1)$ and $Agent(2)$, are active or not, respectively. The n is the number of candidate agents.

For example, when a robot is moving in a corridor and a corner of the corridor is the target, which the Target(corner)-Searcher fails to detect, the strength of the driving command from the Target(corner)-Searcher becomes 0. In that case, the Wall-Follower is the most preferred agent to recover from the failure and is selected if the strength, which is calculated as $\mathbf{v_{i+1}} - \mathbf{v_i}$ in (2.8), is larger than a threshold value.

On the other hand, the Infinite-loop-motion failure recovery rule is described as,

if (find-loop(MH[0], MH[1], ..., MH[i-1]) = TRUE)
  if $V(1) > Th(1)$ Select Agent(1)
    else if $V(2) > Th(2)$ Select Agent(2)

else if $V(i) > Th(i)$ Select Agent(i)

.....

else if $V(n-1) > Th(n-1)$ Select Agent(n-1)

else Select Agent(n)

Here, *find-loop()* is a logical function which becomes *true* in the case that repetition of the same driving command becomes larger than a threshold value. $MH[k]$ is a history of driving commands that is selected at time k and is memorized in a Motion Command Memory. In our system, the maximum value of memory size i is set to 10.

For example, when a robot is going to exit a room, if the Target(exit)-Tracker fails in tracking and a robot repeats wandering for a long period, the value of the find-loop() becomes true. In that case, the Open-Space-Explorer is the most preferred agent to recover from the failure and is selected if the strength, which is calculated using $\mathbf{v_{i+1}} - \mathbf{v_i}$ in (2.8), is larger than a threshold value.

2.5 Summary of issues in the proposed architecture

In summary, the merits of proposed architecture compared with conventional systems are as follows:

1. A robot based on the proposed architecture can robustly navigate itself by using a behavior from the three groups (Reflexive, Purposive Navigation and Adaptive) because the architecture is basically a behavior-based one. The robot can efficiently reach a goal by following a sequence of sub-goal and goal targets, without a detailed geometric map which is hard to prepare.

2. Due to collecting basic behaviors into the three groups, a new behavior can be easily add to the system by just involving it to an appropriate group according to the type of the behavior. Differing from the fixed arbitration in the subsumption architecture, each group has its own arbitration mechanism that can provide

the adaptability to the system against dynamic changing environments.

3. A motion command can be simply determined by just selecting a behavior in the Motion Executor according to the priority of the three groups. No predetermined rules or models in conventional SMPA-based systems are required in advance. In addition, rules for selecting an appropriate behavior in the Adaptive Behavior Agent Group can be installed if necessary.

2.6 Experimental results

Experimental results of both simulation and real navigation showed the effectiveness of the proposed architecture.

2.6.1 Simulation experiments

First of all, the adequate selection of behaviors in the Motion Executor was tested using a simulator[34]. Stationary objects in the environment such as desks or computers are shown as rectangles. The position of the robot is shown as a circle. The heading is indicated as a small line from the center of each circle. The sequences of the circles express the locus of the robot. The mission is to exit the room (enclosed region in the left side of the figure) and to reach the final goal (a black circle in the right side of the figure).

Fig.2.9 and Fig.2.10 show the motion sequence when failure of Purposive Navigation Agent Behavior Group agents occurs in the room space.

Fig.2.9 shows the results *with* the Adaptive Behavior agents in addition to the Reflexive Behavior agents for self-continuance and the Purposive Navigation Behavior agents. In this case, following the Missing target failure recovery rules as shown in the last section, the Open-Space-Explorer in the Adaptive Behavior Agent Group is selected to recover from the failure when the Target(exit)-Searcher cannot find the exit and it guides the robot to the neighborhood of the exit. When the exit appears in the view, the Target-Tracker becomes active and it dominates the robot to exit the room. Even when the robot approaches the

wall, the Wall-Follower in the Adaptive Behavior Agent Group guides the robot to get out of the room by following the wall.

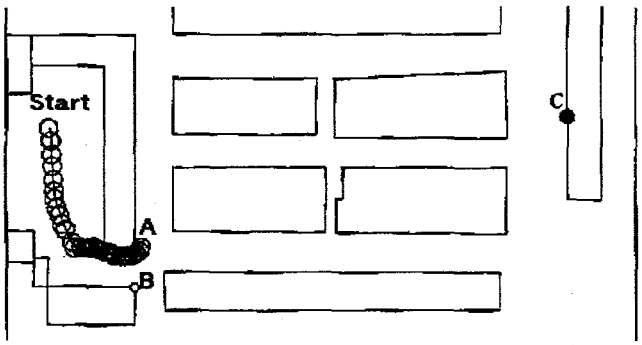


Figure 2.9: Motion sequence with Adaptive Behavior agents. Stationary objects in the environment such as desks or computers are shown as rectangles. The position of the robot is shown as a circle. The heading is indicated as a small line from the center of each circle. The sequences of the circles express the locus of the robot.

Fig.2.10 shows a result *without* the Adaptive Behavior Agent Group agents, that is, only the Target-Searcher, the Target-Tracker in the Purposive Navigation Behavior Agent Group and the Obstacle-Avoider in the Reflexive Behavior Agent Group navigate the simulated robot. Because a static wall stands between the start position and the exit of the room space, Target-Searcher which corresponds to the exit fails. Then the robot starts wandering and goes to collision-free space in front. When the robot approaches the opposite wall, the Obstacle-Avoider becomes active and switches the direction of the robot repeatedly until the exit is in the view, that is, a infinite-loop-motion situation occurs. In several experiments with changing start positions, the robot could not reach the exit of the room space except when the start position is included in the oblique region of Fig.2.10 and the exit is in the view.

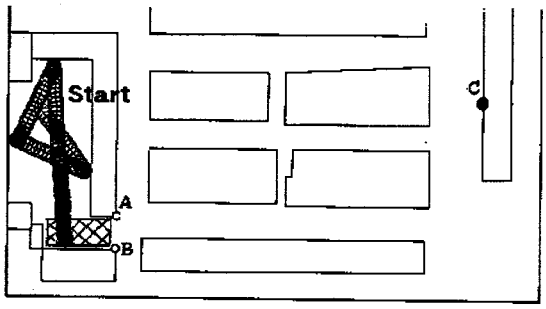


Figure 2.10: Motion sequence without Adaptive Behavior agents. The robot could not reach the exit of the room space except when the start position is included in the oblique region and the exit is in the view.

2.6.2 Experiments on real robot navigation

To evaluate the effect of environmental condition changes, such as, sudden appearance of obstacles, fluctuation of illumination or slip of actuators, the author and Ishikawa developed a small cart-type robot **BIRDIE**(**B**ehavior-based **I**ntelligent **R**obot in **D**ynamic **I**ndoor **E**nvironment)[33] and did navigation experiments in an office.

Eighteen ultrasonic ranging sensors are installed around the robot to acquire the 360-degrees-surrounding depth information at intervals of 20 degrees, which corresponds to the view angle of each sensor. Passive infrared sensors are installed on the front of it to detect thermal obstacles such as humans. Flexible belt-type touch sensors using electrically conducting rubber are attached to the lower part of the robot. A movable stereo camera platform is mounted on top of the robot. A flux-gate compass is located in the center to detect the self-direction. The implementation of the system is distributed in a computer network external to BIRDIE and in an onboard CPU within it. The computer network consists of an engineering workstation and vision systems. Sensor data (sonars, infrared, contact and compass) and a driving command are communicated between the onboard CPU and the network through a wireless serial link. The onboard CPU controls the sensor driver cir-

cuits and motor drive circuits for the camera platform and the wheels. Fig.2.11 shows the overview of the robot.

Real navigation experiments using the robot in an office, which had similar constitutions as simulation experiments, had proved the adequate agent selection in the proposed architecture.

The topological constitution of the office, that is, the sequence of target objects for sub-goals and a goal, were manually taught by an test run.

Fig.2.12 shows an example of agent selection when a human suddenly appears in a way while BIRDIE is approaching a fire extinguisher as a final goal. At first, a Target-Tracker is successfully selected and it is going straight ahead **(a)**. When a human appears in the way of the robot **(b)**, and it is detected by sonar or infrared sensors **(c)**, the O-A agent is selected and the robot starts avoiding behavior **(d)**. After BIRDIE passes the human **(e)**, the Target-Tracker becomes active and the robot starts going ahead to the target again **(f)**.

Fig.2.13 shows a target tracking result sequence, and Fig.2.14 shows a motion sequence of the robot in a corridor. A Target-Tracker navigates the robot to the next intersection. The target (an intersection, in this case) is taught as a pair of vertical edges in front. BIRDIE successfully approaches the intersection. Fig.2.15 shows an example of a real motion sequence. Subgoals are vertical edges of room exit(A,B), turning corners(D,E) and (F,G). The goal is a fire extinguisher(C).

At first, the robot missed the room exit(A,B) because it is out of the view. Then, following the Missing target failure recovery rule, given in the previous chapter, the O-S-E is selected and navigates the robot near the exit. When the robot approaches the exit and the Target(A,B)-Tracker becomes active, the T-T navigates the robot to the exit in the same manner as in the simulation experiments. When the robot intrudes in a corridor, the next Target(D,E)-Tracker becomes active and the robot starts going ahead. When the robot approaches too near a wall and the T-T fails, the W-F becomes active and it navigates the robot near the turning corner(D,E) along the wall. Finally, from (F,G) to (C), the Target(C)-Tracker successfully navigates BIRDIE to the final goal.

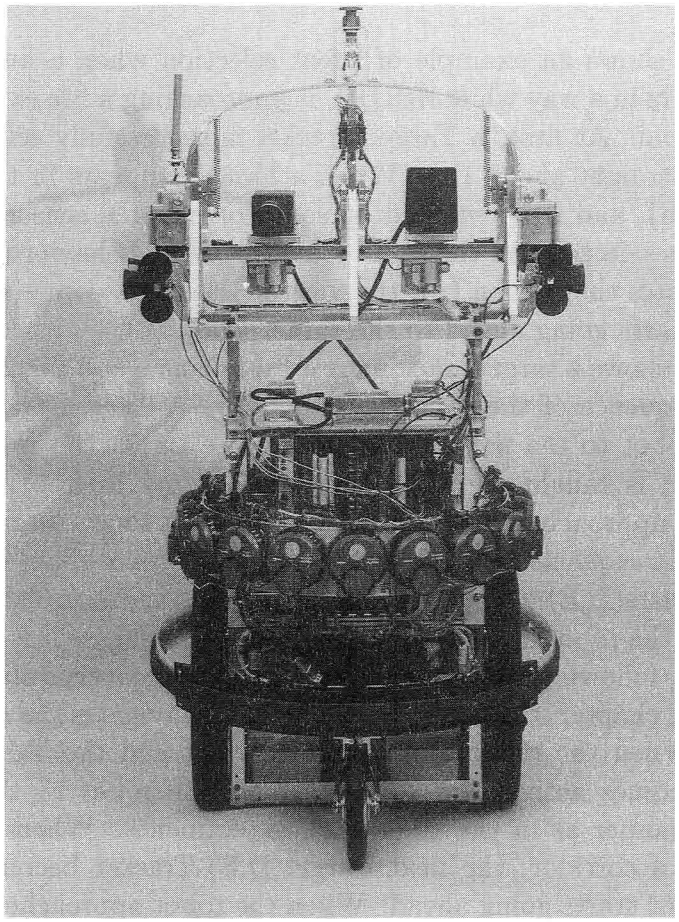
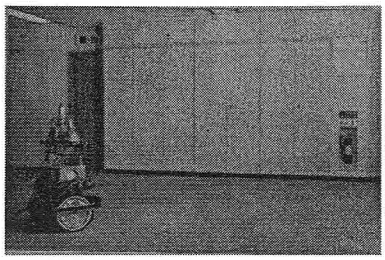
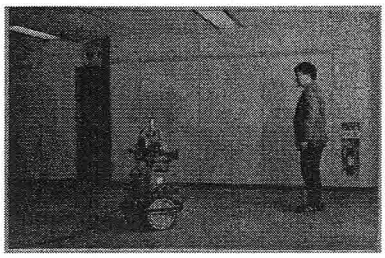


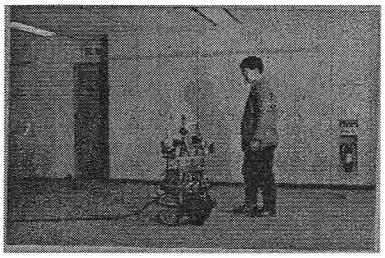
Figure 2.11: BIRDIE overview.



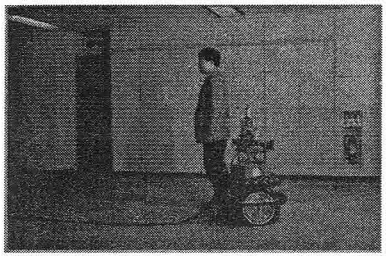
(a) Start moving.



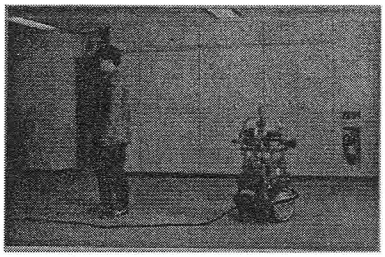
(b) Human appears.



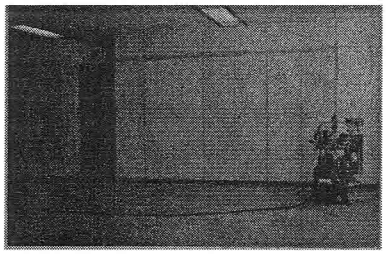
(c) Human is detected.



(d) Start avoiding.

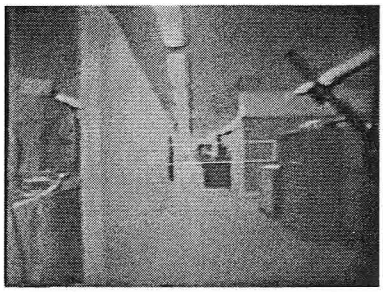


(e) Finish avoiding.

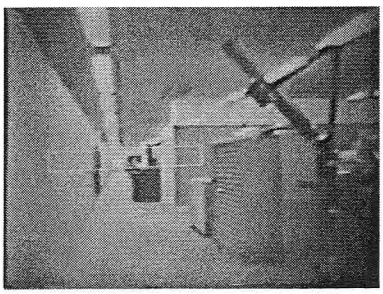


(f) Approach the target again.

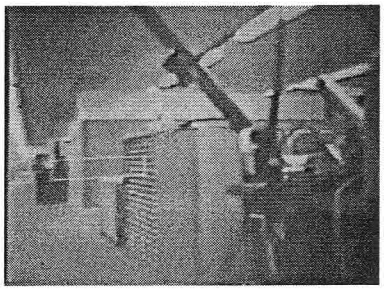
Figure 2.12: Robot motion sequences of obstacle avoidance



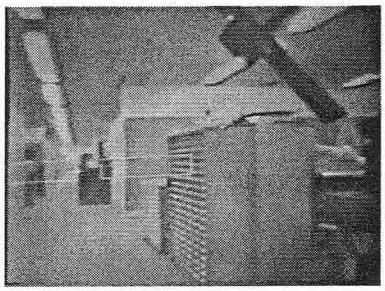
(a) Intersection tracking result (t=1).



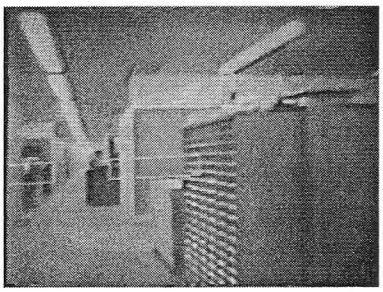
(b) Intersection tracking result (t=2).



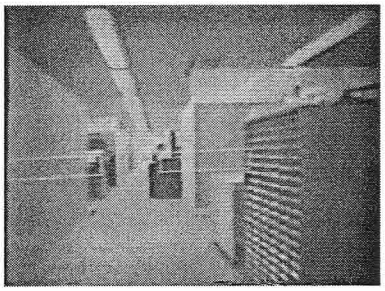
(c) Intersection tracking result (t=3).



(d) Intersection tracking result (t=4).

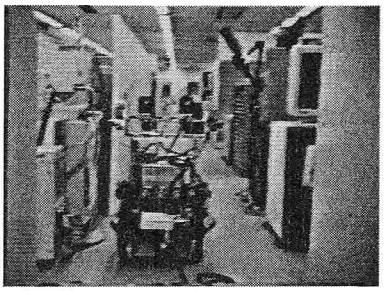


(e) Intersection tracking result (t=5).

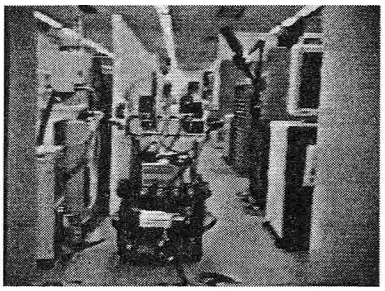


(f) Intersection tracking result (t=6).

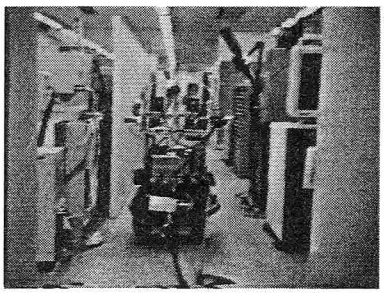
Figure 2.13: Image processing results of the Intersection-Tracker. A square in each figure shows a result of vertical edge tracking as an intersection target.



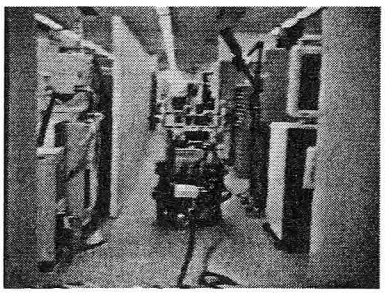
(a) Motion sequence (t=1).



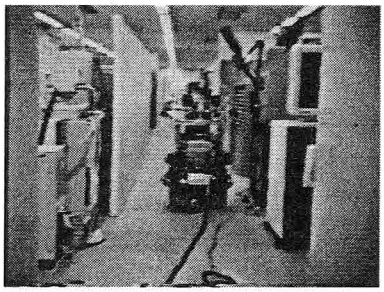
(b) Motion sequence (t=2).



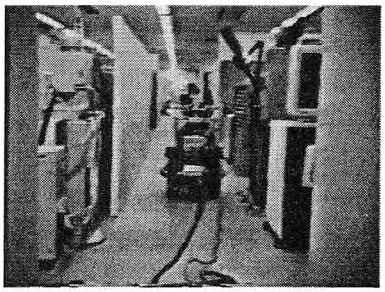
(c) Motion sequence (t=3).



(d) Motion sequence (t=4).

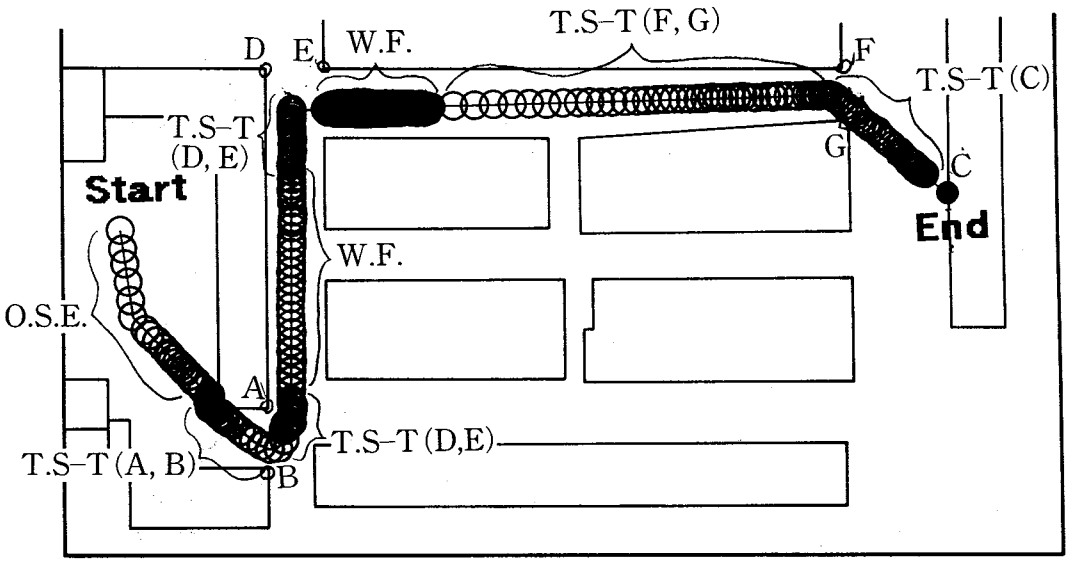


(e) Motion sequence (t=5).



(f) Motion sequence (t=6).

Figure 2.14: Robot motion sequences of passing along a corridor.



T.S:Target-Searcher,T-T:Target-Tracker,
O.S.E.:Open-Space-Explorer, W.F.:Wall-Follower

Figure 2.15: Total robot motion example. The mission is to exit the room (enclosed region in the left side of the figure) and to reach the final goal (a black circle in the right side of the figure). Subgoals are vertical edges of room exit(A,B), turning corners(D,E) and (F,G). The final goal is a fire extinguisher(C). The sequences of the circles express the locus of the robot.

Chapter 3

Free space detection by stereo vision

From the viewpoint of robustness only, the most reliable sensor is a contact-type sensors such as a rubber sensor for detecting obstacles; however the detection is only done when an obstacle is touched by the robot, that is, the range of obstacle detection is very small. Most robot researchers are using ultrasonic and laser ranging sensors because they provide an inexpensive means to obtain range information around a robot by computing the echo travel time.

The main drawbacks of ultrasonic ranging sensors are as follows:

1. They easily fail to detect an obstacle whose surface is inclined and has specular reflection property, because the power of reflection echo becomes weak.

2. They just detect the existence of obstacles.

   Recognition of obstacle types is generally difficult from ultrasonic ranging information.

Vision sensors, on the other hand, can provide rich information for recognizing obstacles and are more reliable for detecting specular objects, even though they are less reliable than contact-sensors and ultrasonic ranging sensors in situations in which obstacles are occluded

or illumination conditions change. That is, a vision sensor is necessary for intelligent mobile robots to attain the adaptive capability and intelligence for global recognition.

In this chapter and the next chapter, vision-based obstacle detection algorithms are explained.

3.1 Related work

3.1.1 Geometry of stereopsis methods

As for detecting static obstacles, stereo-based methods have been proposed to build visual maps for collision-free space extraction.

The geometry of binocular stereo vision is shown in Fig.3.1[38].

A given point P is projected to a point I_1 in image 1, then its match point I_2 in image 2 is searched. Point I_2 necessarily belongs to a straight line DE_2 of image 2 determined completely by I_1, which is called the *epipolar line* associated with I_1.

Let C_1 and C_2 be the center of cameras 1 and 2, respectively. Then, the 3D position of P is uniquely determined as the cross point of the two lines I_1C_1 and I_2C_2. Epipolar lines DE_1 and DE_2 are cross lines of the plane which includes the points C_1, C_2, and P, and, image 1 and image 2, respectively

3.1.2 Related work on passive stereopsis methods

As for detection of correspondences in stereo images, numerous algorithms have been proposed. They can roughly be classified into two categories.

- Area-based approaches

  In these approaches, the system attempts to correlate the grey levels of image patches in the views being considered, assuming that they present some similarity. The resulting depth map can then be interpolated.

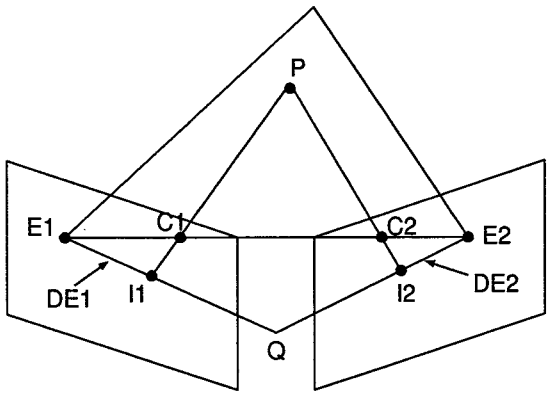


Figure 3.1: The geometry of binocular stereo vision.

The underlying assumption appears to be a valid one for relatively textured areas; however, it may prove wrong at occlusion boundaries and within featureless regions.

Alternatively, the map can be computed by directly fitting a smooth surface that accounts for the disparities between the two images. This is a more principled approach since the problem can be phrased as one of optimization; however, the smoothness assumptions that are required may not always be satisfied.

- Feature-based approaches

  These algorithms extract features of interest from the images, such as edge segments or contours, and match them in two or more views.

  These methods are fast because only a small subset of the image pixels are used, but may fail if the chosen primitives cannot be reliably found in the images; furthermore, they usually only yield very sparse depth maps.

In general, instead of *points*, *structural features* such as vertical edges are extracted from stereo images and matched using geometric constraints[39], or knowledge-based constraints[40][41].

Ayache proposed a method for making a three-dimensional model of an environment where a mobile robot is working, by using stereo vision[40]. To decrease the uncertainty of noisy stereo measurement, the extended Kalman filtering technique is utilized for building, describing and fusing visual maps among various viewpoint positions. He also proposed a use of the trinocular stereo vision to make stereo matching reliable[42].

Nakayama, Yamaguchi, Shirai and Asada proposed a reliable feature-based multi-stage stereo matching method which determines more reliable pairs of matching edges earlier than less reliable ones and makes use of previous matching results[43]. The contrast of an edge segment is used as the measure of reliability of the match because a high contrast edge has high positional and directional accuracy. The decisions as to whether there are correspondences between pairs of lower contrast edges or not becomes easier, because previously found more reliable matching results are available to reduce the occurrences of ambiguous matches.

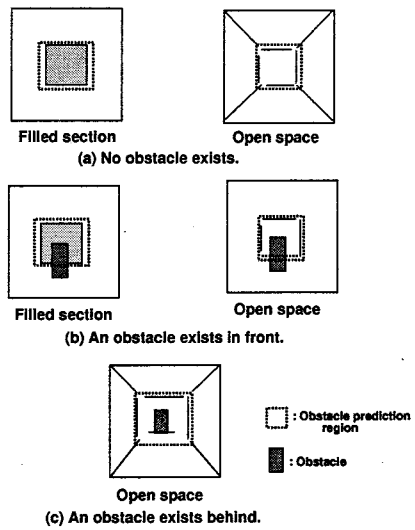


Figure 3.2: Examination of obstacle existence situations.

The author, Onoguchi, Kuno, Hoshino and Tsunekawa proposed a method to examine the following three statuses by checking the intensity sum of an obstacle prediction region, which is surrounded by stereo matching edges[41].

1. There is no obstacle across the space where an existence of obstacle is being examined as shown in Fig.3.2(a).

2. An obstacle exists in front of the space as shown in Fig.3.2(b).

3. An obstacle exists behind the space as shown in Fig.3.2(c).

The matching at an occlusion boundary is difficult, because the contrast of an edge segment at an occluding boundary may drastically change where the depth gap is large due to difference in the view direction of cameras.

To solve this problem, Nakayama and Shirai proposed a feature-based matching method which enables matching of occlusion boundaries using normalized light intensity[44]. At first, light intensity is normalized so that intensities derived from matching edges, which are obtained from texture edges, are similar between images. Next, edges at occlusion boundaries are matched using the normalized intensity. Experimental results using synthetic images and real images showed the effectiveness of the method.

3.1.3 Related work on generation of environment description

As for representation of an environment, several methods have been proposed.

An *octree* can represent an environmental space by recursively subdividing a region into octants. A tree node is labeled *black* if it is completely contained within an object, labeled *white* if it is completely contained within a free space; otherwise, the node is labeled *gray*. Using this representation, a path planning problem can be solved efficiently, such as finding the successive adjoining white octree cells from a starting cell to a final goal cell [45].

Occupancy grids proposed by Elfes[46] is a three-dimensional cell representation to integrate various outputs from multiple sensors, such as ultrasonic sensors and TV cameras.

The occupancy grid is a multidimensional random field that maintains stochastic estimates of the occupancy state of the cells in a spatial lattice. To construct a sensor-driven map of the robot's world, the cell state estimates are obtained by interpreting the incoming range readings using probabilistic sensor models. The Bayesian estimation procedures allow the incremental updating of the occupancy grid using readings taken from several sensors over multiple points of view.

The *artificial potential field* approach has been widely studied for robot path planning[47]. In the approach, a repulsive force is given at an obstacle position and an attractive force is given at a final goal position to create a potential field in physical sense. An appropriate path can be found by threading valleys of the potential field from a start position to a final goal position.

Three-dimensional *wire-frame representation* is also proposed [48] by applying the Delaunay triangulation method for stereo measurement results.

Recently, *the proximity space method*[49] was proposed to perform real-time, behavior-based control of visual gaze and obstacle avoidance. *The proximity space* was a limited three-dimensional region of space where the existence of objects was examined. The examination was done, by matching surface texture of objects on the PRISM-3 stereo vision system[50], and combining the result with motion measurements. The performance of this method mainly depended on the texture information of objects as same as that of the area-based stereo methods.

3.2 Disparity Prediction Stereopsis Method

In order to achieve safe navigation, a depth map for a whole scene is not necessary. All that is required is information about the type and the existence of obstacles to detect collision-free space.

The author, Onoguchi, Kuno and Asada have proposed a practical collision-free space detection method (**Disparity Prediction Stereopsis (DPS) Method**)[51][52] for robots working in nuclear power

plants. Stereo vision was used to extract the region whose disparity range corresponding to the navigation space by feature matching. Fig.3.3 shows the composition of the DPS Method.

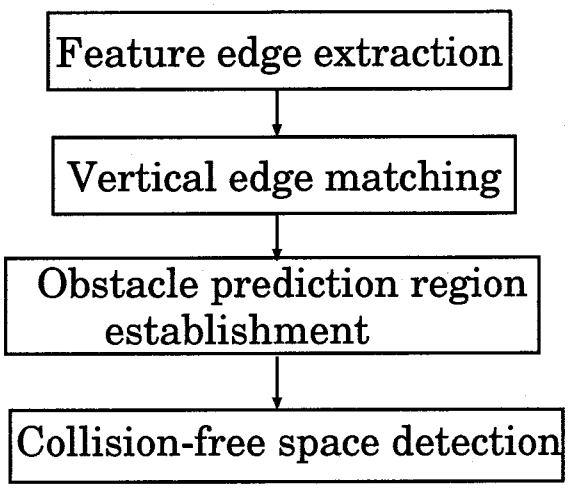


Figure 3.3: Composition of the Disparity Prediction Stereopsis Method.

3.2.1 Prediction of disparity and feature edge matching

First, the space, where the robot is due to move, is established as shown in Fig.3.4, and the disparity range that corresponds to the position and the space size is calculated.

Let the viewpoint height of the robot be X, the velocity of robot progress be V and the processing time be T. The search space of size $X \times V \times T$ is defined at the front distance of $V \times T$ from the present position. The objects existing in this space are the obstacles to be detected. Then, the search space is divided into a certain number of cross sections which are vertical to the camera axis, and the disparity values of objects which may exist on the cross section are calculated. If an obstacle exists in the space, the following status is generally realized:

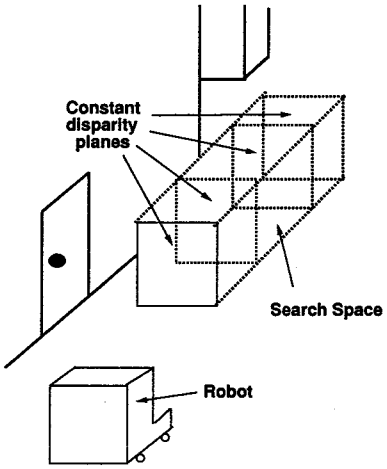


Figure 3.4: Establishment of search space.

Vertical edge segments that are candidates of standing obstacle contours appear in the space.

Next, edge segments are enhanced and vertical ones are extracted from stereo camera images whose optical axes are set parallel.

Fig.3.5 shows the algorithm for extracting feature edges from images. The reasons why feature edges are used for matching, instead of raw grey level data are the stability against the difference in γ-characteristics of two cameras and the lower computational cost.

First, the regions with high spatial discontinuity values are enhanced by processing the 3×3 SOBEL filter. Next, the images are transformed to binary data by predetermined threshold value, processed by thinning operation, and independent components are labeled to measure the length values. Then, the components with small length are eliminated as noises, and other components are tracked by measuring the local curvature values to describe edge segments. Finally, the components which consist of vertical edge segments are selected for initial matching.

Fig.3.6 shows the configuration of a stereo camera system. Let the centers of the stereo camera lenses be $O1$ and $O2$, and the projections of a point P be $P1$ and $P2$, respectively. When the stereo cameras

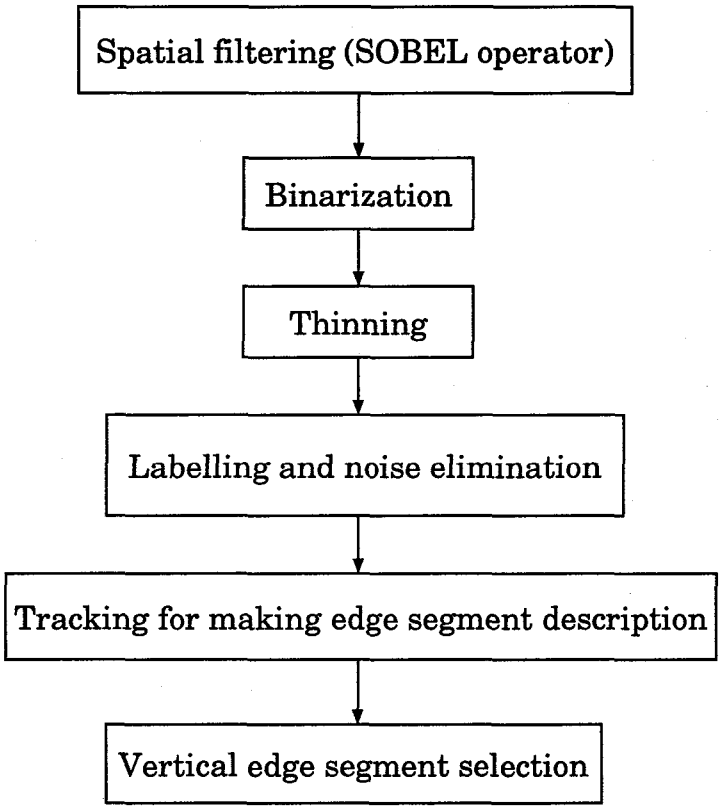


Figure 3.5: Algorithm of feature edge extraction.

are set at the same height, and the optical axes of stereo cameras are parallel, the relationship between the distance and the disparity S is simply given ($S = S1 \sim S2 = \frac{E \times A \times F}{D}$) as shown in the figure. Here, A is an interval between two cameras, F is focal length, E is distance/pixel conversion ratio and D is distance between stereo camera and an object.

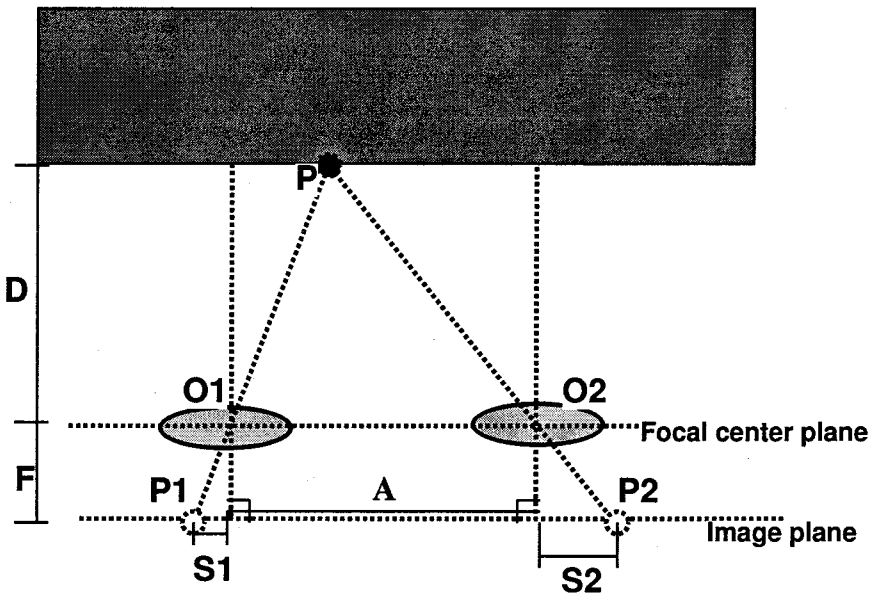


Figure 3.6: Stereo camera configuration. When the stereo cameras are set at the same height with the optical axes of stereo cameras parallel, the relationship between the distance and the disparity S is simply given ($S = S1 \sim S2 = \frac{E \times A \times F}{D}$).

Next, the extracted edge segments are matched among a base mask along the edge segment in a left image and prediction masks which are set in a right image with the disparity range, as shown in Fig.3.7. By combining the feature-based matching and the intensity-based matching in these ways, matching accuracy can be improved.

If the discovery of a matched pair of vertical edges implies that there exists an obstacle on a cross section, then the match is examined for the remaining edges. If there is no matched pair, it is judged that no

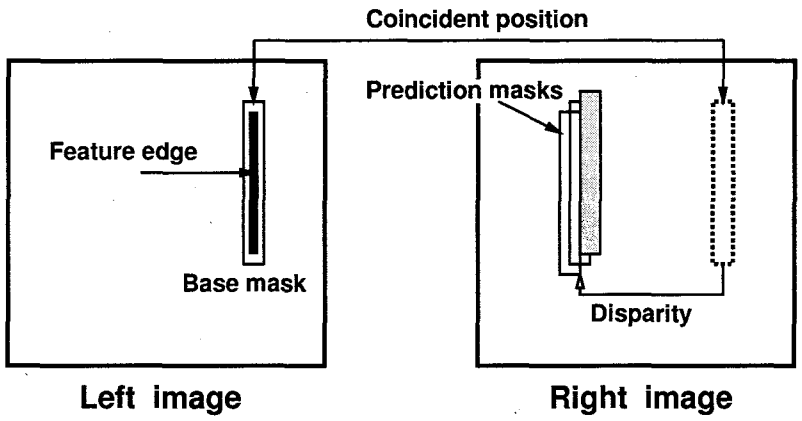


Figure 3.7: Edge segment matching by disparity prediction. The extracted edge segments are matched among a base mask along the edge segment in a left image and prediction masks which are set in a right image with the disparity range.

object exists on the cross section, and the matching in the next cross section is examined. Thus, the existence or absence of objects in robot moving space is determined based on the vertical edge matching.

By using the DPS Method, an intelligent Static Obstacle Avoider agent in the Reflexive Behavior Agent Group for self-continuance can be realized so that the agent becomes active and provides the stopping behavior when matched edge segments are extracted during navigation.

3.2.2 Extraction of passage region

If the view directions of stereo cameras are parallel to a floor, when an object does not interfere with the space where the robot is moving, as in the case that the object is far and high above the viewpoint, an object on a front wall is projected on an image with its lower end in a region above a passage region. To check this status, the following heuristic conditions which are useful for extracting passage regions are built in the system.

- Spatial continuity:

  A connected region with a large area in the lower part of an image is assumed as a candidate for passage region.

- Existence of roadside edge:

  A long edge in the lower part of an image is assumed as a candidate for the border of passage region.

- Absence of vertical edge:

  An object standing on a floor obscures a farther passage. Consequently, when the view direction is parallel to the floor, a region between vertical edges is more likely to belong to a standing obstacle. This assumption does not apply when the passage has a texture, as in a case of a floor with lattice stripes. It is almost always valid for a monochrome floor and can be used effectively in the passage detection.

- Uniformity:

  In a region adjacent to a passage, intensity characteristics, such as average intensity value, color and texture, do not greatly change. This assumption is not true in general when a floor is uneven, has shadow or reflection.

  In practical use in indoor situations, a two-dimensional passage detection method is efficient from the viewpoint of low computational cost, because a part of passage region is usually involved in an image from a high camera position of a robot.

  The above conditions are checked for each of the local regions in the scene. According to the validity of the conditions, the experimentally determined scores are given to pixels in the region. The result is considered to be the reliabile representaion of a local region which is passage region.

  Fig.3.8 shows the passage extraction algorithm. Using the conditions (b) and (c) in the extraction of passage region, it is decided whether or not an edge is contained in a local region. If an edge is not contained, the condition (d) is used to calculate the average and the variance of the intensity in the region. If the results

do not differ greatly from the reference values, which have been calculated for the specified sample regions in the scene, a score is given. This process is applied to the lower half of the scene, from the bottom upward. The reason for this is that when the optical axes of stereo cameras are parallel to the floor, the passage is only projected to the lower half of an image.

When the robot is moving, the average value in the previous process is used as the reference value continuously. Finally, the region whose total score is higher than the threshold value, which is set experimentally, is extracted. Using the condition (a), the connected region with the maximum area is extracted as a passage region. Among edge segments matched by the predicted disparity, those with the lower ends contained, or in contact with the above passage region are interpreted as the candidates for obstacles.

3.2.3 Measurement of obstacle position

For the moving robot to plan a route to avoid the detected obstacles, the position of the obstacles in the world (environment) coordinate system is calculated, using

(1) the position of the lower end of the obstacle edge,

(2) the distance between the camera and the edge determined from the disparity at the match, and

(3) the self-location measurement result of the robot itself, which is built into the robot as a part of the visual navigation system shown in Fig.1.2.

Fig.3.9 shows the schematic diagram of measuring obstacle position. Let the distance and the direction of the stereo cameras in regard to the passage border be L and G, respectively. Let the distance between the camera and the obstacle determined from the disparity prediction be D, and the deviation of the obstacle from the optical axes be K. Then the distance W between the passage border and the obstacle is given by the following formula:

$$W = L - D \sin G - K \cos G \tag{3.1}$$

where L and G are given by the self-location measurement subsystem.

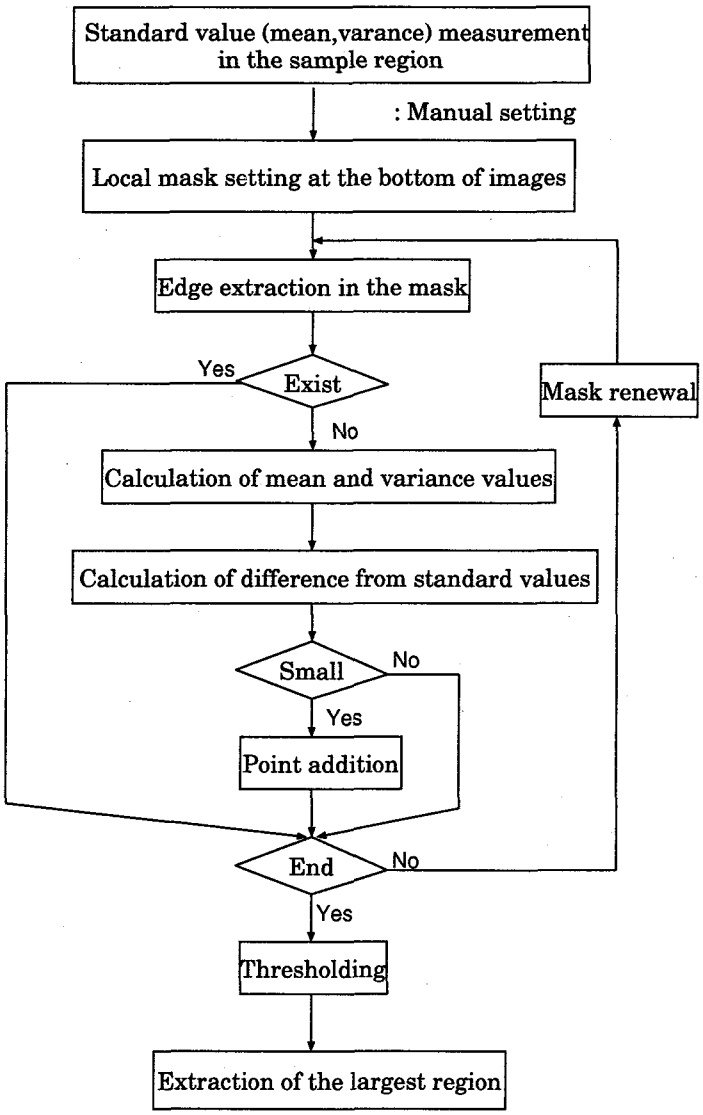


Figure 3.8: Algorithm of passage extraction.

K is determined by distance between the image center and the lowest end point of the matched edge segment connected with the matched vertical edges and multiplying the distance by $\frac{D}{F}$, where F is the focal length.

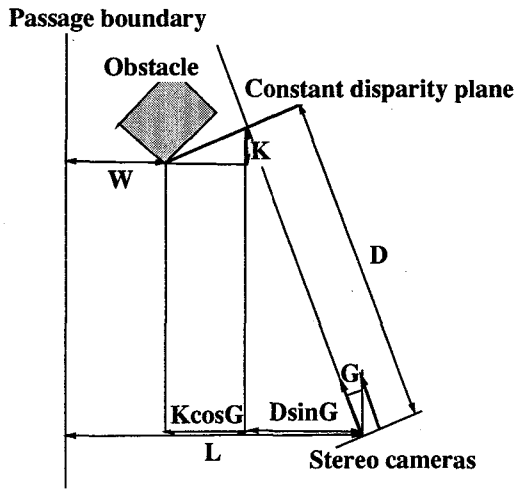


Figure 3.9: Measurement of obstacle position.

Fig.3.10 shows the total flowchart of the DPS method described above.

3.3 Experimental results

To evaluate the performance of the DPS Method in regard to the complexity in shape, obstacle detection experiments using both five artificial straight and curved objects (cubes, cylinders, triangular cones, circular cones and tori) and humans have been done.

The DPS software was implemented on a workstation (Toshiba AS4260) with an additional hardware TOSPIX-Π[53] for image processing and edge segment description. As the stereo cameras, two CCD cameras with $560 \times 350(196,000)$ pixels (Ikegami FCK-10) were employed. The focal length of lenses was 12.5mm and the distance between

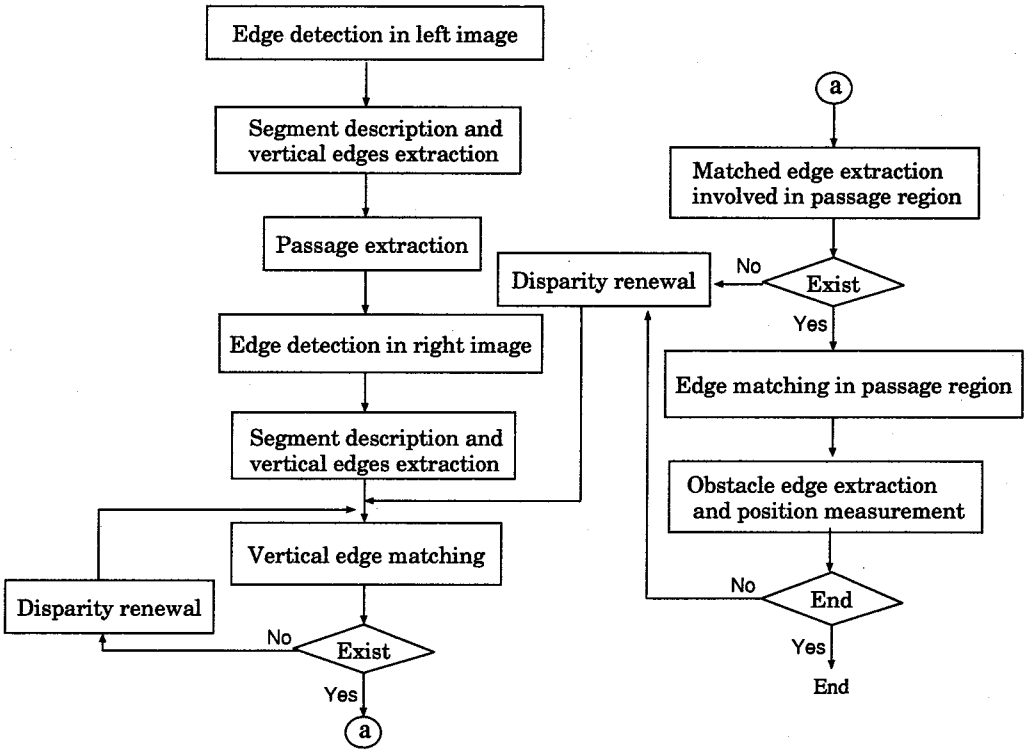


Figure 3.10: Detailed flowchart of the DPS method.

two cameras was 20.0cm.

The obstacles were set to stand on a tile floor with a uniform color (green) and the distance ranges from stereo cameras were varied from 2.0m to 4.0m by 50.0 cm. The experiments have been done five times for each object and each distance range (120 times in total) by randomly varying the positions of the obstacles.

According to the extraction status for the matched edge segments contained in passage region, the results are divided into five cases as follows:

1. Both ends of the obstacle are successfully detected without false matches.

2. One end of the obstacle fails to be detected without false matches.

3. Both ends of the obstacle are detected with an edge other than the detected obstacle (false matches occur).

4. One end of the obstacle fails to be detected, and an edge other than the detected obstacle is detected (false matches occur)

5. The obstacle wholly fails to be detected.

Table 3.1 shows the results. A denominator in each cell shows a total trial number(5) and a numerator shows a successful trial number.

The error of detecting only one side of object boundary (case-2) occurred four times, that is, once for a torus (350cm to 400cm) and three times for humans (300cm to 400cm). The main reasons for missing in detecting one end of obstacles were that edge segments of curved boundary part at far distance were extracted disconnected due to low intensity, because the stereo cameras approached too close to vertically installed pipes and entered shadow). Missing in detecting obstacles caused by false matching did not occur.

The success rate of perfect detection was 96.7%.

Figs.3.11 $\sim$ 3.15 show examples of the obstacle detection results. In an obstacle detection result, a left original image, a passage extraction result and detected obstacle edge segments involved in the passage region are shown overlapping. Boundary edges of objects involved in

Table 3.1: Experimental results of collision-free space detection using Disparity Prediction Stereopsis Method.

	200~250cm	250~300cm	300~350cm	350~400cm
cube	5/5	5/5	5/5	5/5
cylinder	5/5	5/5	5/5	5/5
circular cone	5/5	5/5	5/5	5/5
triangular cylinder	5/5	5/5	5/5	5/5
torus	5/5	5/5	5/5	4/5
human	5/5	5/5	4/5	3/5

passage regions were successfully detected for all cases, while detection of passage regions partially failed due to low-contrast and shadows as shown in Fig.3.11. In many cases, only edge segments in passage regions were detected because feature edges were divided to independent segments at the boundary of passage region, and the edge matching and the examination of interaction with passage regions were done independently. In the case of humans (Fig.3.14, Fig.3.15), the borders of feet, which are contained in passage regions, are extracted successfully. The processing time was 7 seconds on average.

Next, obstacle detection experiments in an environment simulating a nuclear power plant were executed.

The passage was composed of a tile floor painted green, as in the experiments described above. The background contained a power distribution box and measuring instruments. In addition to the background complexity, the scenes were affected greatly by the reflection from the floor and shadows cast by other objects. Consequently, a false match may frequently occur and make it difficult to detect obstacles with certainty. The obstacle was an orthogonal box placed 3 to 4m in front of the stereo cameras, which corresponded to the disparity from 90 to 65 pixels.

Here, two examples are shown. In the first case, an obstacle (cube) is involved in both stereo images. In the second case, the right side of the cube in the left image is occluded.

Fig.3.16 and Fig.3.20 show both original and edge images.

In the first case, the total numbers of edges in the left and right images were 674 and 481, feature edges were 59 and 72, edge segments were 2458 and 1780, and vertical edge segments for initial matching were 33 and 33, respectively.

In the second case, the total numbers of edges in the left and right images were 724 and 531, feature edges were 70 and 73, edge segments were 2479 and 1859, and vertical edge segments for initial matching were 28 and 28, respectively.

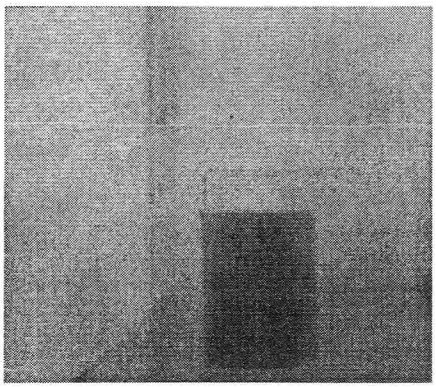
Fig.3.17 and Fig.3.21 are the results of vertical edge matching in the cross sections containing an obstacle with the disparity from 90 to 65 pixels by 3 pixels shift. At this stage, a false match, at a region involving the power distribution box, occurred.

Fig.3.18 and Fig.3.22 are results of extracting passage regions. The passage was extracted almost correctly, except for miss-detection of part of the box caused by disconnection of the lower edge due to shadow effect.

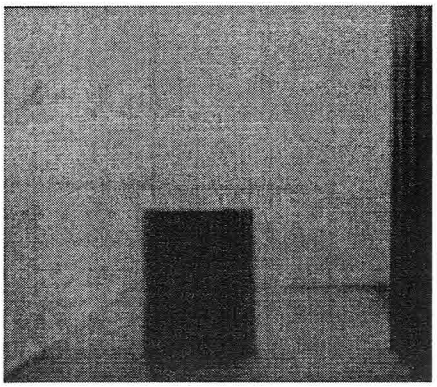
Finally, Fig.3.19 and Fig.3.23 are the edges detected as obstacles at the final stage, overlapped with the left image. The false match of the power distribution box was eliminated because it was not be contained in passage region, and only the both boundaries of the box were successfully detected.

In another experiment, when the box was placed nearer and its right side was out of left view, the left boundary was successfully detected. The processing time for those cases was approximately 10 seconds.

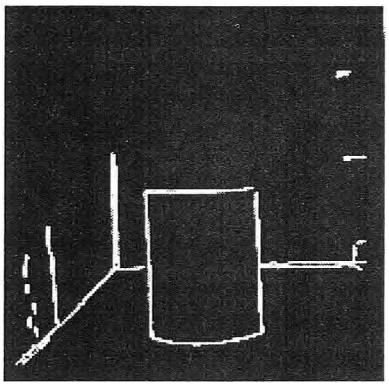
From real navigation experiments using the clover-type vehicle AIMARS developed by Toshiba Corp., shown in Fig.3.24, to which stereo cameras were attached, obstacles in the scene were reliably detected 110 times and stable navigation at a speed of 500 m/h could be achieved.



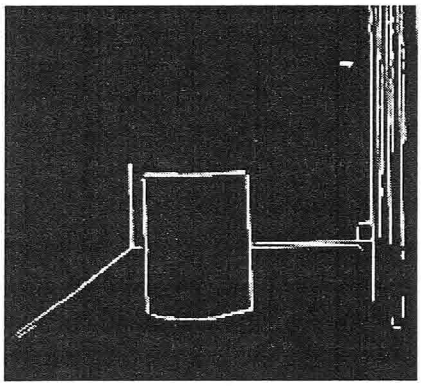
(a) Left original image



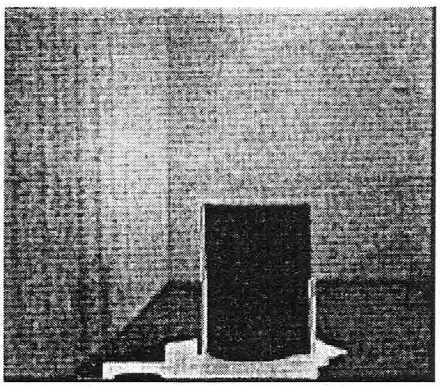
(b) Right original image



(c) Left edge image

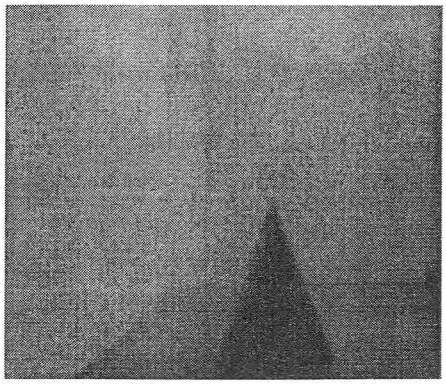


(d) Right edge image

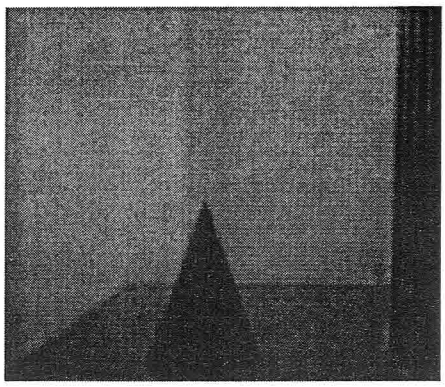


(e) Obstacle detection results

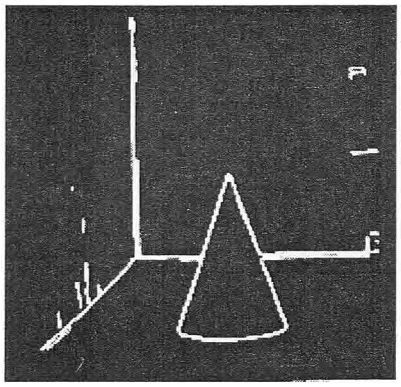
Figure 3.11: Obstacle (cylinder) detection example. Edge segments detected as obstacles are shown as white lines in (e). White region below the cylinder is a result of passage detection process.



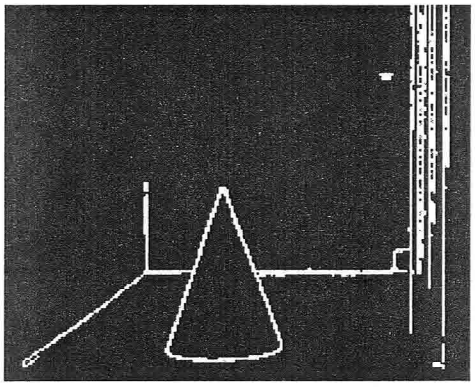
(a) Left original image



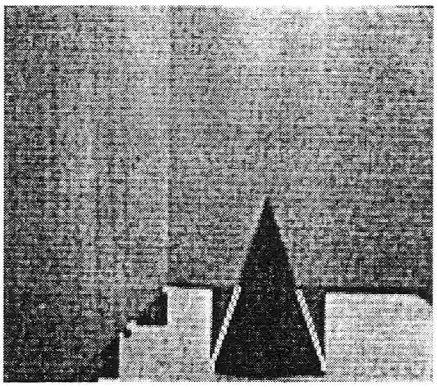
(b) Right original image



(c) Left edge image

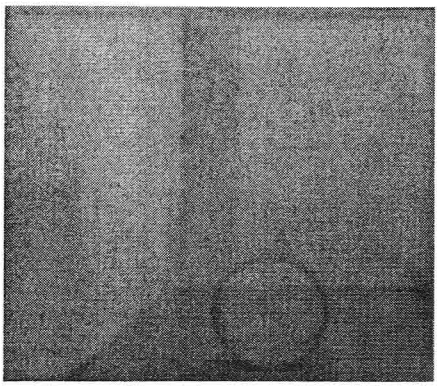


(d) Right edge image

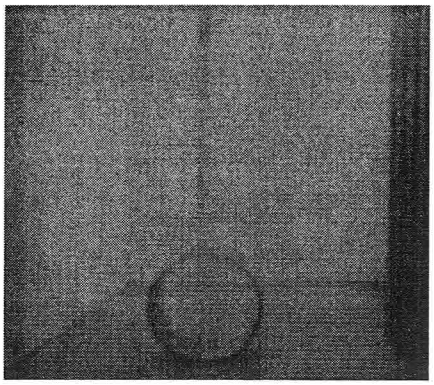


(e) Obstacle detection results

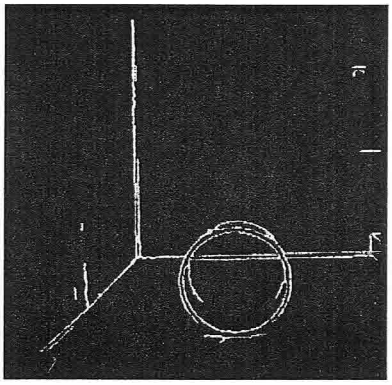
Figure 3.12: Obstacle (circular cone) detection example.



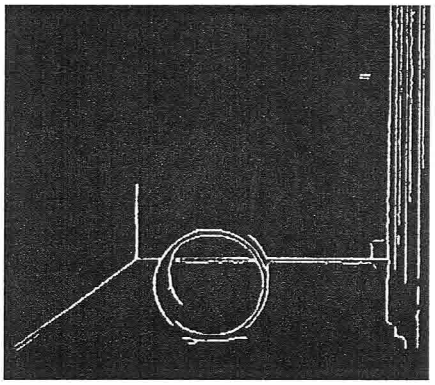
(a) Left original image



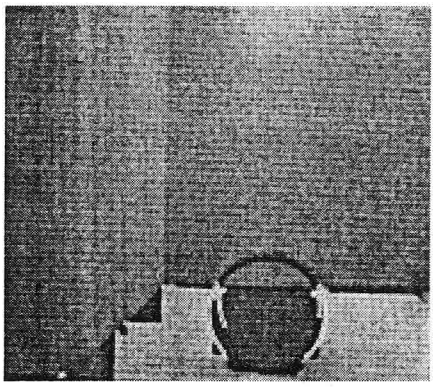
(b) Right original image



(c) Left edge image



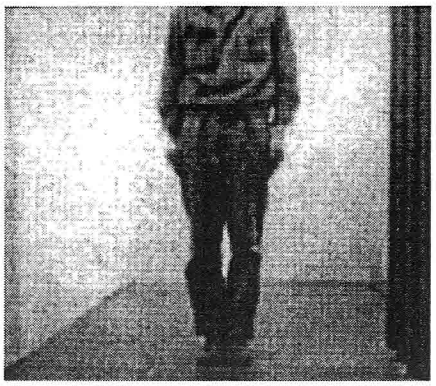
(d) Right edge image



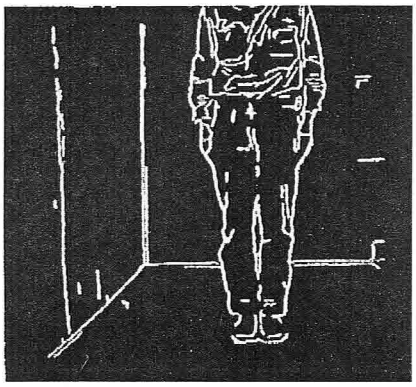
(e) Obstacle detection results

Figure 3.13: Obstacle (torus) detection example.

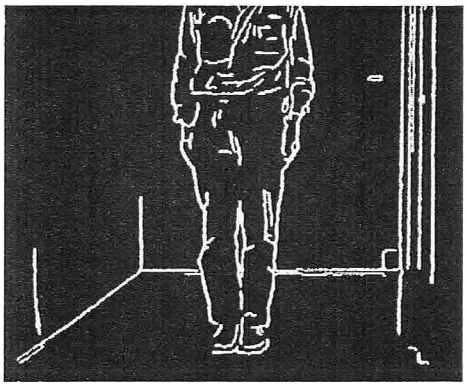
(a) Left original image



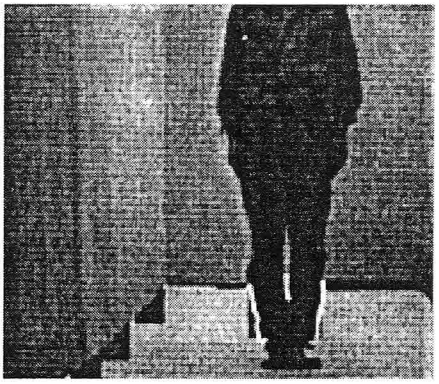
(b) Right original image



(c) Left edge image



(d) Right edge image

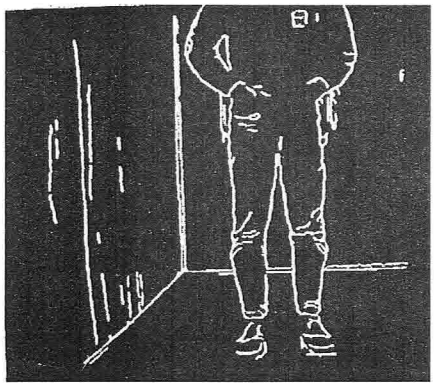


(e) Obstacle detection results

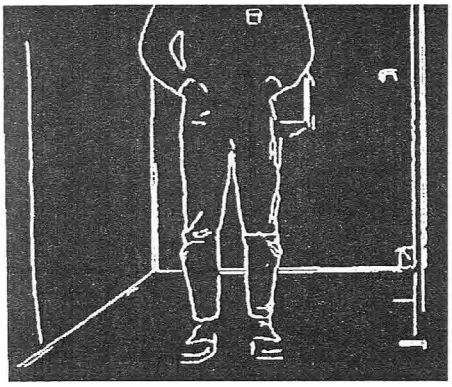
Figure 3.14: Obstacle (human) detection example No.1.

(a) Left original image

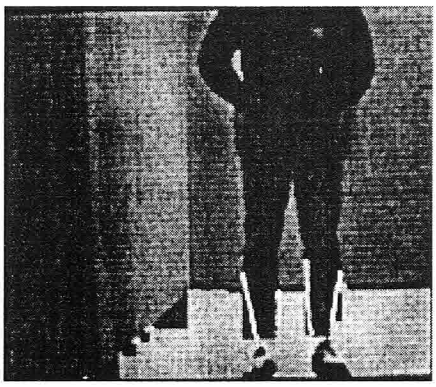
(b) Right original image



(c) Left edge image



(d) Right edge image



(e) Obstacle detection results

Figure 3.15: Obstacle (human) detection example No.2.

3.3.1 Discussions

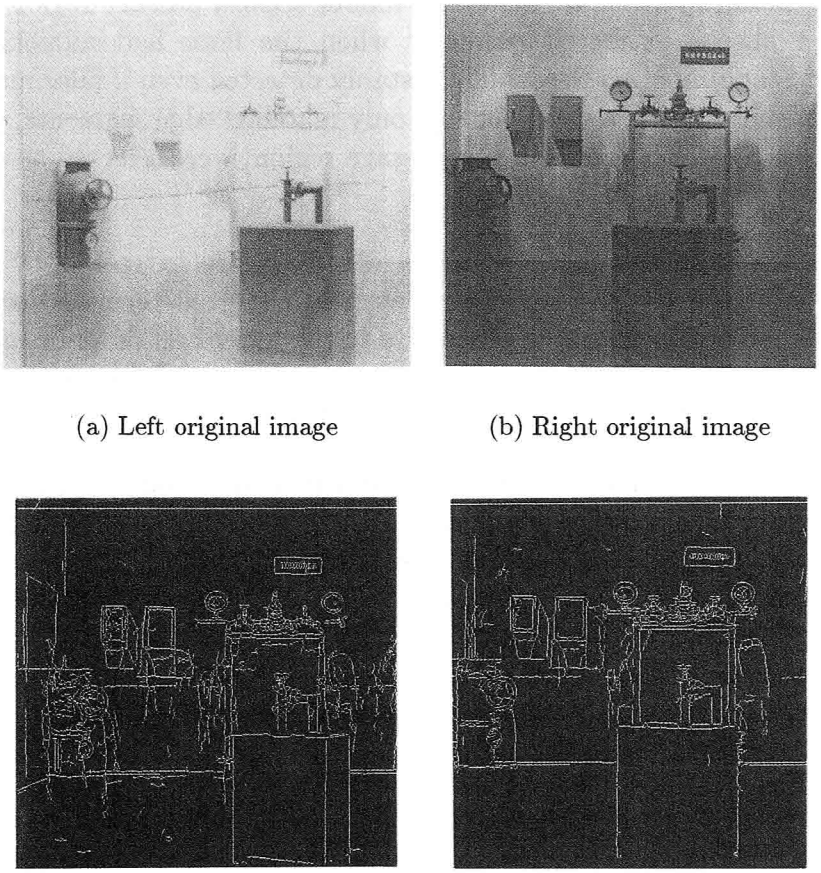
From these experimental results in indoor scenes, passage regions were almost always extracted correctly when the floor had monochrome color. Standing obstacles could be stably detected even if false matches occurred in front of a wall, because only matched edge segments, whose lower ends were involved in the passage region, were selected as candidates of obstacles.

In the proposed method, passage regions are extracted by using the conditions set in a heuristic way, and the predetermined score is added. After applying this procedure to the lower half of an image, the threshold processing is applied. Then the connected region with the maximum area is defined as the passage region.

This method is useful when the floor is monochrome and obstacles are located at a relatively long distance, as in the series of experiments executed in this study. However, the detection becomes difficult when a robot approaches obstacles and occlusion of floor by obstacles becomes large. In addition, the probability of incorrect obstacle detection increases when a floor contains a high contrast shadow or texture. To solve this problem, the use of time continuity and the use of height information have been examined.

It is not always true that an object, in the space where a robot is moving, is contained in passage regions. When an obstacle is close and the bottom part is out of view, or it is located higher from a floor, the lower part may be projected in the upper half of an image. To manage this problem, an improvement of the system considering the view angle is necessary so that the search space is set at a sufficiently long distance in front, and detection and matching are executed continuously.

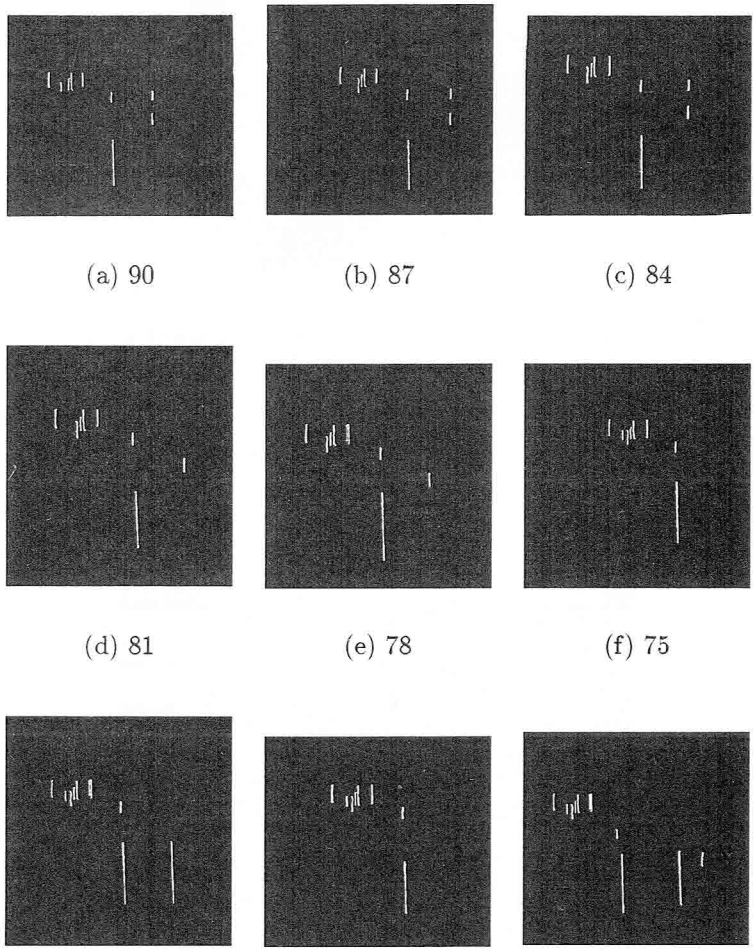
The matching time for one cross section by the DPS method was approximately 0.02 seconds using a standard engineering workstation. On the other hand, a long processing time is necessary for extracting passage regions. It is necessary to reduce total processing time in order to achieve real-time navigation.



(a) Left original image (b) Right original image

(c) Left edge image (d) Right edge image

Figure 3.16: Obstacle detection in simulated nuclear plant example No.1 : original and edge images. The total numbers of edges in the left and right images were 674 and 481, respectively. The number of vertical edge segments used in the matching process were 2458 and 1780, respectively.



(a) 90 (b) 87 (c) 84

(d) 81 (e) 78 (f) 75

(g) 72 (h) 69 (i) 65

Figure 3.17: Vertical edge matching example No.1 for a cross section.

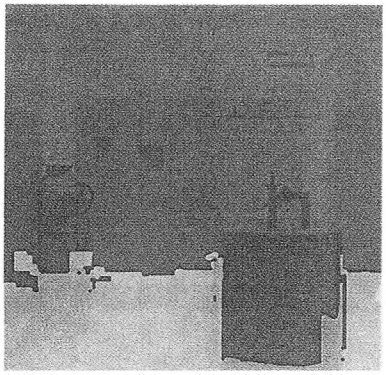


Figure 3.18: Obstacle detection in simulated nuclear plant example No.1 : passage region extraction results. White region shows the result of passage extraction.

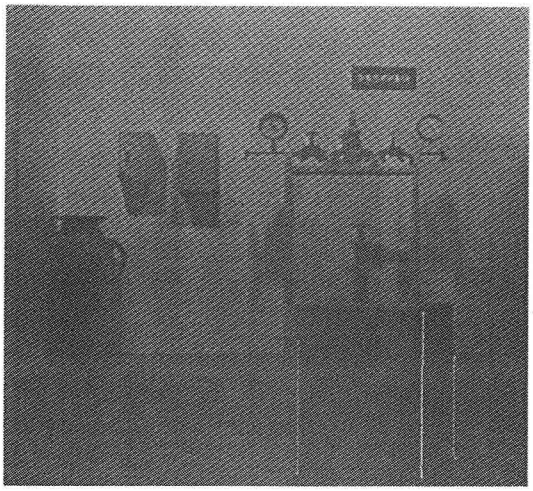
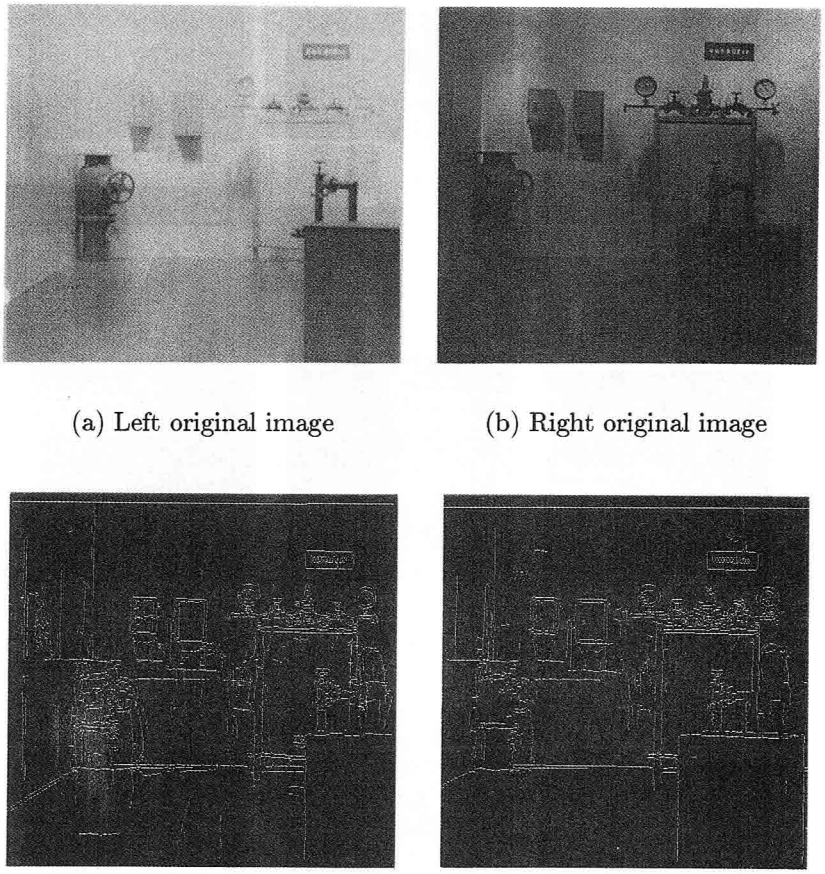


Figure 3.19: Obstacle edge detection result example No.1. Detected obstacle edges are shown as white lines.



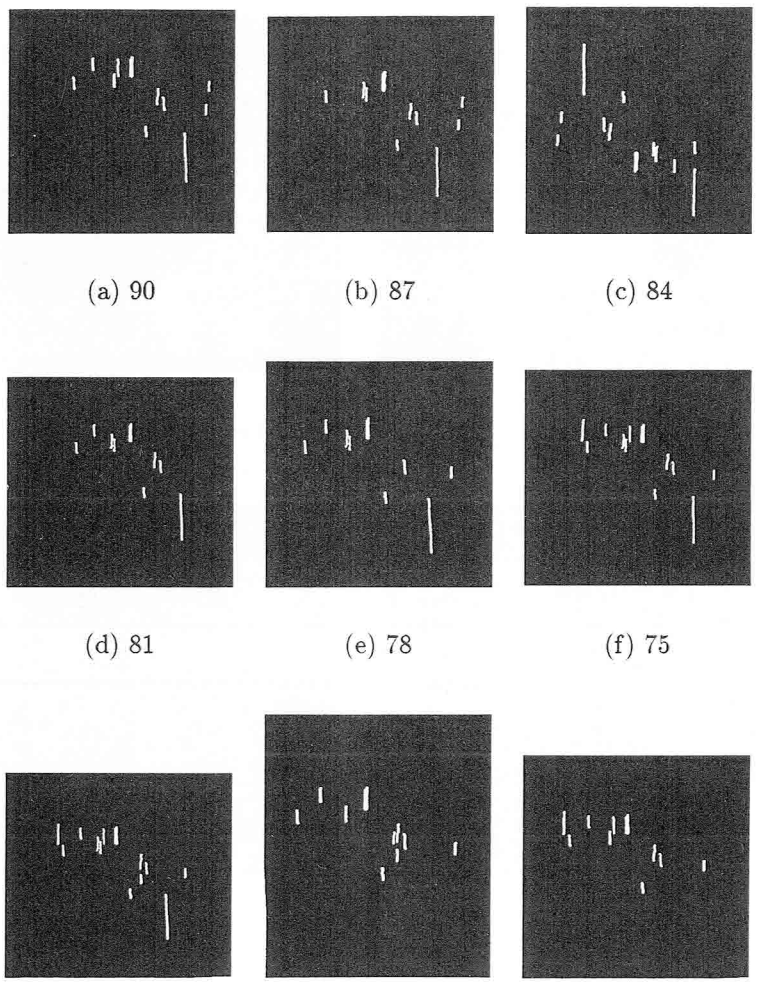
(a) Left original image

(b) Right original image

(c) Left edge image

(d) Right edge image

Figure 3.20: Obstacle detection in simulated nuclear plant No.2 : original and edge images. The total numbers of edges in the left and right images were 724 and 531, respectively. The number of vertical edge segments used in the matching process were 2479 and 1859, respectively.



(a) 90 (b) 87 (c) 84

(d) 81 (e) 78 (f) 75

(g) 72 (h) 69 (i) 65

Figure 3.21: Vertical edge matching example No.2 for a cross section.

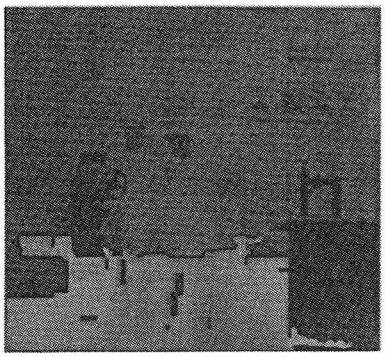


Figure 3.22: Obstacle detection in simulated nuclear plant example No.2 : passage region extraction results. White region shows the result of passage extraction.

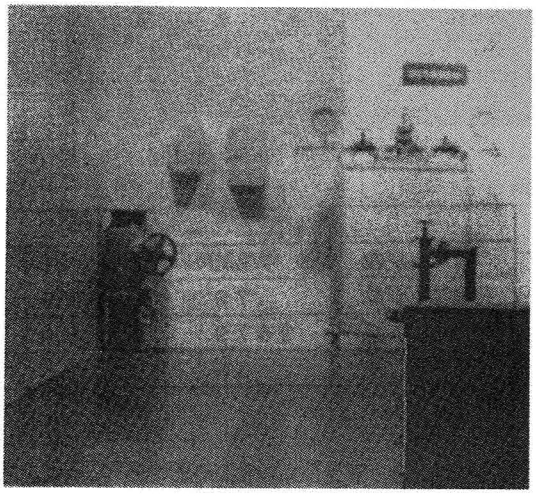
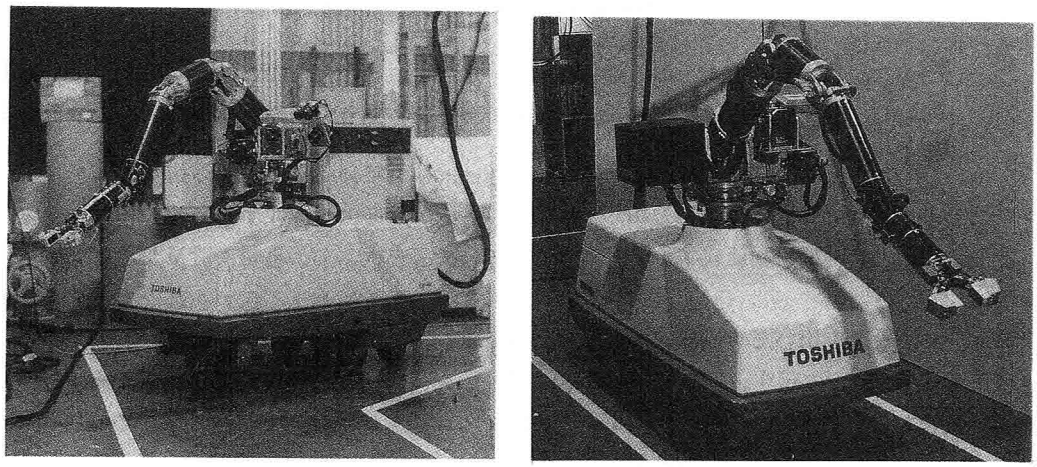


Figure 3.23: Obstacle edge detection result example No.2. Detected obstacle edges are shown as white lines.



(a) Motion sequence (t=1)

(b) Motion sequence (t=2)

Figure 3.24: The overview of AIMARS.

3.4 Planar Projection Stereopsis Method

The above-mentioned problems regarding the extraction of passage regions, Onoguchi, the author and Takeda. have proposed a road region extraction method, the **Planar Projection Stereopsis (PPS) Method**[54] which extracts collision-free space on a road by using height information.

Since a road area can be assumed to be a sequence of flat planes in front of a vehicle, the height information is useful for extracting a road area.

By using the PPS Method, the decision, whether each point in stereo images exists on the road or not, can be easily done.

At first, PPS calculates a planar equation representing a road area by using height and pose of stereo cameras on a vehicle.

Next, stereo images are projected to the plane, where corresponding points are projected to the same positions on a certain road area if they really exist on a road plane as shown in Fig.3.25, while corresponding points with different heights from the road plane are projected to different positions in each stereo image as shown in Fig.3.26.

A Planar Projection Description is obtained by a subtraction between projected images from a set of stereo images and a road area can be represented by a set of points with small values.

Experimental results for real road scenes have shown the effectiveness of the PPS method as shown in Figs.3.27 $\sim$ 3.28. By checking whether parts of edges detected by the DPS Method are involved in a region detected by the PPS method or not in image planes, only obstacles on a ground passage can be effectively extracted.

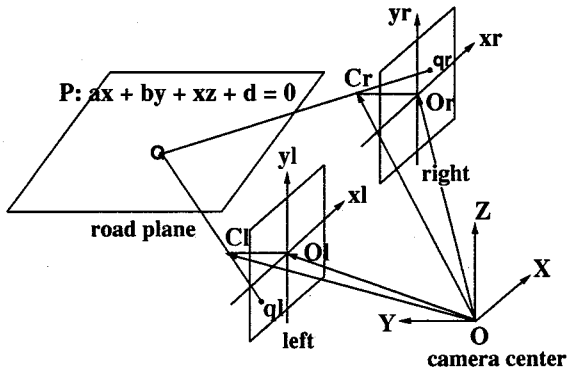


Figure 3.25: Projection of a point on a road plane. Stereo images are projected to the plane, where corresponding points are projected to the same positions on a certain road area if they really exist on a road plane.

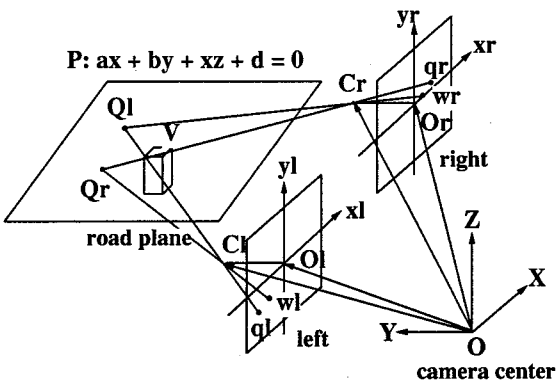
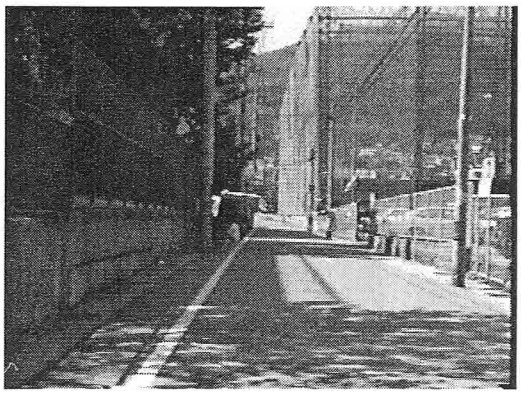
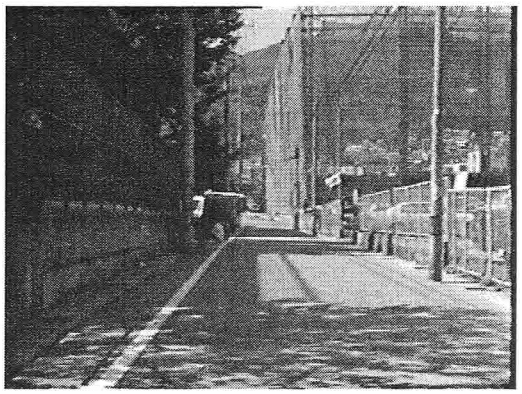


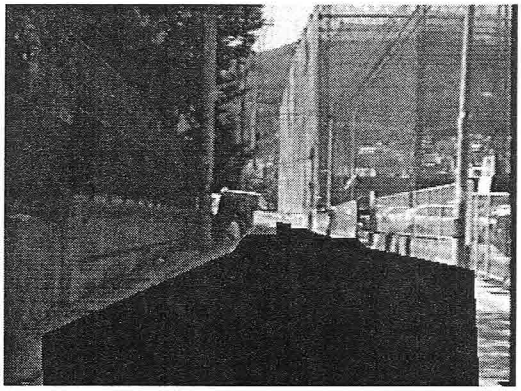
Figure 3.26: Projection of a point whose height is different from a road plane. Corresponding points with different heights from the road plane are projected to different positions in each stereo image.



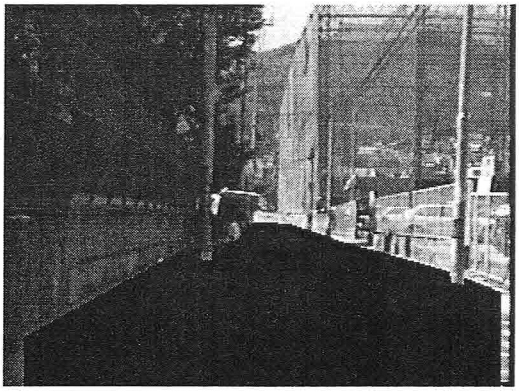
(a) left image



(b) right image

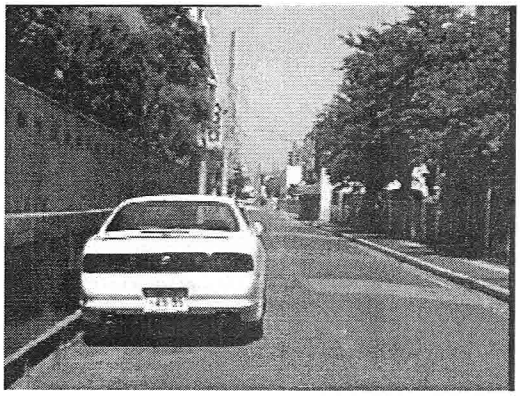


(c) Results of PPS method

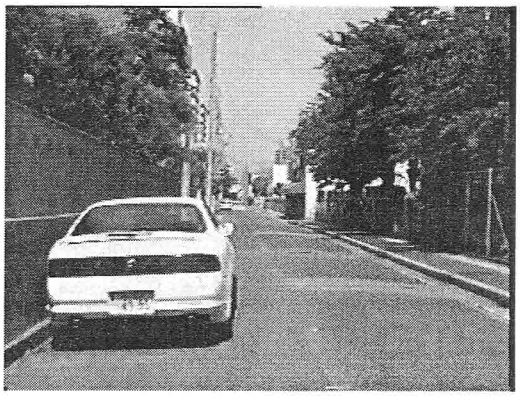


(d) Manually detected road region

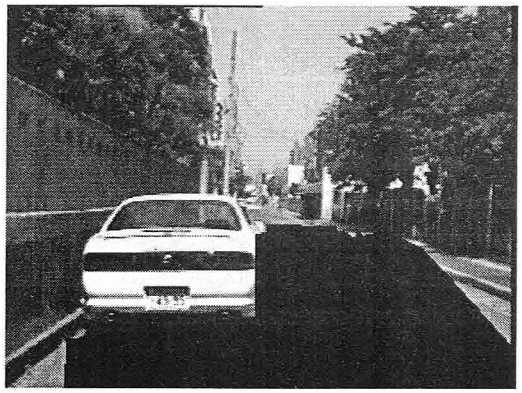
Figure 3.27: Experimental result of PPS method No.1.



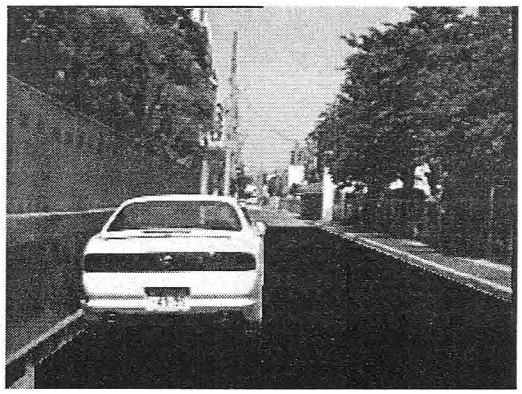
(a) left image



(b) right image



(c) Results of PPS method



(d) Manually detected road region

Figure 3.28: Experimental result of PPS method No.2.

Chapter 4

Moving object recognition by monocular vision

In the case of static obstacles, the existence of an obstacle in a space, where a robot is due to move, is the minimal necessary information for route establishment, as described in the previous chapter, because it does not move and stays in the same position. On the other hand, in the case of moving obstacles in the navigation environments, not only the obstacle detection but also the obstacle recognition processes are required for achieving intelligent avoidance, because the prediction of route interference between a robot and the obstacle is necessary for achieving intelligent obstacle avoidance capability.

In this chapter, obstacle detection and recognition method **MOROFA** by analyzing optical flow information is described.

First, optical flow field is detected in image sequences from a camera on a moving observer and moving object candidates are extracted by using a square residual error value that is calculated in the process of estimating the FOE (Focus of Expansion).

Next, the optical flow directions and intensity values are stored for pixels involved in each candidate region, to calculate the distribution width values around the principal axes of inertia and the direction of the principal axes.

Finally, each candidate is classified into a category of object that is expected to appear in the scene, by comparing the proportion and the direction values with standard data ranges for the objects which are

determined by preliminary experiments.

Experimental results of car/bicycle/pedestrian recognition in real outdoor environments have shown the effectiveness of the method.

4.1 Related work

- Obstacle detection

  Conventional vision-based moving obstacle detection methods are classified into two groups: those which *do not use* motion analysis and those which *use* motion analysis.

  The former group is further divided into two groups; that is, *stereo-based method group* mentioned in the previous chapter in which just an existence of obstacles is detected without distinction of moving obstacles and static obstacles, and *tracking-based method group* in which a moving obstacle is detected as a temporal or spatial difference in a local window and tracked to examine if the change is really due to an obstacle or not.

  In the VaMoRs vechicle[55], local windows were set in image planes to detect obstacle edges. When some edges were detected, they were tracked continuously as edges of passing vehicles. A real-time vision system BVV3 was developed for the purpose [56].

  In the case of stereo-based detection, segmentation process among static objects(background) and moving obstacles is necessary, though the process is difficult to achieve and computationally expensive.

  In the case of tracking-based detection, false tracking frequently happens due to various intensity patterns such as, pedestrian crossing marks or shadows of parking cars, that result in detection errors.

  On the other hand, in the latter motion-based obstacle detection group, region segmentation process is generally required before detection.

  Adiv has developed an optical flow field segmentation method[57] to extract independent moving objects by applying Hough transformation methods to optical flow fields. The approach is based

on two main stages. In the first stage, the flow field is partitioned into connected segments of flow vectors, where each segment is consistent with a rigid motion of a roughly planar surface using a Hough transformation. In the second stage, segments are grouped under a hypothesis that they are induced by a single, rigidly moving object. Each hypothesis is tested by searching for three-dimensional motion parameters which are compatible with all the segments in the corresponding group. The advantage of the Hough transformation is the robustness against noise in optical flow fields. On the other hand, incorrect segmentation leads to inaccurate motion estimation, that induces failure of obstacle detection. In addition, enormous memory is necessary for the Hough transformation with multiple parameters, where five unknown parameters exist for *structure from motion (SFM)* problem.

Tian and Shah presented a method to determine 3D motion of multiple objects from two perspective views[58]: In the method, segmentation is determined based on the 3D rigidity constraint. First, an input image is divided into overlapping patches, and for each sample of the translation parameter space, the rotation parameters of patches are computed using least-squares fitting. Every patch votes for a sample in the translation parameter space. For a patch containing multiple motions, an M-estimator is used to compute rotation parameters of a dominant motion. The Adaptive Hough Transformation method is used to refine the relevant parameter space in a *coarse to fine* fashion. By using this method, an optical flow field can be reasonably segmented, and a memory burden can be greatly reduced by using the Adaptive Hough Transformation, because a small accumulator is used with the iterative *coarse to fine* accumulation and search strategy. However, its calculation cost is very expensive because a camera motion has to be estimated in every local image patch and the estimation is carried out by iterative operations.

Ohta has proposed a method which uses inconsistency and estimation errors computed in estimating the SFM[59]. Using the assumption that a region of moving obstacle is sufficiently small

in an image, a region which satisfies at least one of these three conditions is extracted as a moving obstacle. The conditions are,

(1) square error value of the SFM process is large,

(2) estimated distance to an object has negative sign, and

(3) estimated distance is large.

This method requires camera motion parameters, though they are difficult to estimate. In addition, precise optical flow estimation results are necessary for the SFM process.

- Object recognition

  Conventional motion analysis research on obstacle avoidance has been focused on the detection of obstacle existence and only a few attempts have been done for moving object recognition.

  For object recognition, most conventional methods are shape-based, that is, those methods have used the features related to shape of the objects.

  In the ACRONYM system[60] for recognizing airport scenes, *ribbons description* which are 2-D projections of 3-D models is generated from extracted edge segments. Then, a generalized cylinder description is generated to match the 3D shape model for airplanes. ACRONYM is one of the most successful model-based vision systems. The main problem with ACRONYM, however, is that it takes a long time to match features extracted from an image with a 3-D model.

  Kuno, Okamoto and Okada presented an efficient object recognition system which automatically generates a recognition strategy from a 3D shape model, and recognizes the object using this strategy[61]. In the system, the appearances of an object from various viewpoints are described with visible 2-D features, such as parallel lines and ellipses. Then, the features in the appearances are ranked according to the number of viewpoints from which they are visible. The rank and the feature extraction cost for each feature are used to generate a tree-like strategy graph. The system searches for features in the order indicated by the graph.

After detection, the system compares the line representation generated from the 3-D model and the image features to localize the object.

There are three common problems for shape-based recognition.

The first problem is the *difficulty for non-rigid objects*, such as a human whose shape changes with time, because an enormous possibility of 2-D feature combinations exists for non-rigid objects. In addition, it is very difficult to create complex models, such as a non-rigid human shape model.

The second problem is the *robustness against occlusions.* If a part of an object is occluded by other objects, the recognition becomes unstable because false features easily occur at the boundary of these objects.

The third problem is the *computational cost.* Conventional systems consist of complex processes such as the model creation process, feature extracting process, description generation process and model matching process. When the shape of the object to be recognized becomes complex, the computational cost increases rapidly.

From the viewpoint of psychological physics, recognition using motion information has been studied for many years.

For example, Johansson showed that a human could recognize the shape and motion of a human in a short time using discrete moving light displays (MLD)[62]. A typical way of producing an MLD is to attach small glass bead reflectors to a person's major joints (shoulders, elbows, wrists, hips, knees or ankles) and focus a strong light on the person. From a sequence of images involving these few discrete points, a human can recognize a pedestrian, and describe the type of motion, such as walking backward, jumping, or walking left. Complicated scenes, such as several independently moving bodies and couples dancing, can be recognized[63].

The velocity field, that represents the motion of object points across an image, is called the optical flow field. Optical flow

results from relative motion between a camera and objects in the scene.

Conventional optical flow estimation methods have been classified into three groups[64].

The first group consists of methods using differential methods (gradient-based)[65][66], that compute optical flow velocity from spatio-temporal derivatives of image intensity, or filtered versions of the image (using low-pass or band-pass filters).

The methods involved in the second group use correspondences (matching technique)[67] as the stereopsis method, that finds the best match position for each image point where a similarity measure, such as the cross-correlation, becomes maximum, and estimates the flow vector as the positional difference between the original point and the best match point.

The third group of optical flow estimation methods is based on the energy outputs of velocity-tuned filters[68], or phase outputs of band-pass filters[69].

The performance measurements of these methods were presented in [70].

As for motion-based moving object recognition, Yasutomi, Mori and Kiyohiro proposed a pedestrian detection method using rhythm information of walking[71]. In this method, rhythms and strides of walking are measured by using change of intensity in a local window of differential images between successive frames. The measurement is robust against variation of clothes, physique and hair styles because the local window is set where a moving object touches a ground. However the recognition ability is limited to pedestrians from a static camera, because a periodic intensity change is frequently observed from a moving observer (for example, periodic pedestrian crossing patterns on a road).

Nelson and Aloimonos developed a qualitative obstacle detection method[72] by using the directional divergence of the 2-D optical flow fields. The process is done following the steps:

1. Creation of partial maps on the basis of gradient direction, and computation of spatial and temporal derivatives.
2. Computation of the parallel component of optical flow for each partial map.
3. Temporal accumulation of information by combining separately computed estimates of the flow at several closely spaced times.
4. Approximation of the directional divergence from partial flow maps in each principal direction.
5. Combination of directional divergence maps to produce a hazard map.
6. Segmentation of an image into three regions: *safe*(magnitude of value in the hazard map is less than the expected error of the computation), *danger*(magnitude of value in the hazard map is greater than the expected error of the computation) and *caution*(no divergence can be computed due to lack of information in the hazard map).

Though the process can be implemented in a highly parallel, layered architecture, the acquired results are only a /danger/safe/caution segmentation.

Camus, Cooms, Herman and Hong implemented the algorithm on a single ordinary UNIX workstation without the benefit of real-time operating system support, and navigated a robot in a laboratory at a speed of 20cm/h for as long as 26 minutes without collision[73].

4.2 Object recognition by optical flow analysis

Our main aim was the development of an algorithm that could efficiently detect moving objects in a scene from a moving observer.

Though several tracking-based approaches have been studied[74], their applications are limited in the case of single (camera) motion in

a scene. On the other hand, the problem to solve here is: *Detect and recognize a moving independent object which appears in front during navigation of a moving observer.*

A possible direct method to solve the above problem is to separate an optical flow field into multiple regions with inherent optical flow patterns and find regions that do not belong to the background, as shown in several related work referred to in the last section.

The main problem in using this method is the weakness against errors in optical flow fields. In real situations, considerable errors are involved in results of optical flow estimation. As a result, boundary detection of inherent moving regions, and estimation of 3D motion parameters from optical flow vectors involved in a local region, are hard to achieve. In addition, the flow segmentation process requires a lot of computation time.

The author, Takeda and Onoguchi propose a recognition method MOROFA (Moving Object Recognition by Optical Flow Analysis)[75] which can detect and recognize moving objects from a moving observer by analyzing optical flow information acquired from dynamic images.

Fig.4.1 shows the block diagram of the MOROFA method.

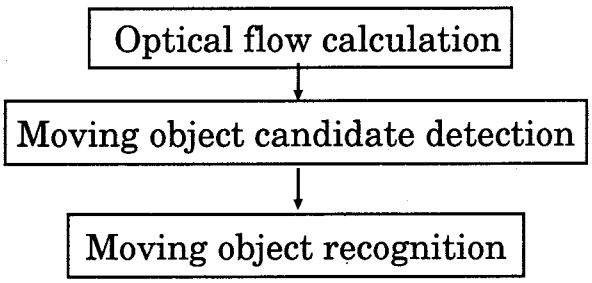


Figure 4.1: Block diagram of the MOROFA method.

First, optical flow field is detected in image sequences from a camera on a moving observer and moving object candidates are extracted by using a square residual error value that is calculated in the process of estimating the FOE.

Next, the optical flow directions and intensity values are stored for pixels involved in each candidate region, to calculate the distribution width values around the principal axes of inertia and the direction of the principal axes.

Finally, each candidate is classified into a category of object that is expected to appear in the scene, by comparing the proportion and the direction values with standard data ranges for the objects which are determined by preliminary experiments.

The MOROFA method is robust against errors occurring in the optical flow fields and in detection of moving object candidates. This method can be applied to many industrial areas in addition to intelligent mobile robots (for example, an intelligent machine surveillance system), because few parameters need to be adjusted for recognition.

4.2.1 Optical flow detection

We used the improved gradient-based method developed by Weber and Malik[76] because it showed the highest efficiency in preliminary experiments.

The flow vector $\mathbf{v}$ is calculated by solving the simultaneous equation shown below, which is composed of multi-scale space-time filtering outputs.

$$\mathbf{A} \cdot \mathbf{v} + \mathbf{I}_t = \mathbf{0} \qquad \mathbf{A} = \begin{bmatrix} I_{1x} & I_{1y} \\ I_{2x} & I_{2y} \\ \vdots & \vdots \\ I_{nx} & I_{ny} \end{bmatrix}, \quad \mathbf{I}_t = \begin{bmatrix} I_{1t} \\ I_{2t} \\ \vdots \\ I_{nt} \end{bmatrix} \tag{4.1}$$

Here, I_{ix} is the output of i−th space-time filter and $\mathbf{v} = (u, v)$ is the optical flow vector to be calculated.

The solution of (4.1) is given as follows:

$$\mathbf{v} = -\left(\mathbf{A}^T\mathbf{A} - \lambda_3^2\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{I}_t \tag{4.2}$$

Here, the parameter λ_3 is the smallest singular value of the measurement matrix $[\mathbf{A}|\mathbf{I}_t]$. In the present system, five space-time filters with three scales are used.

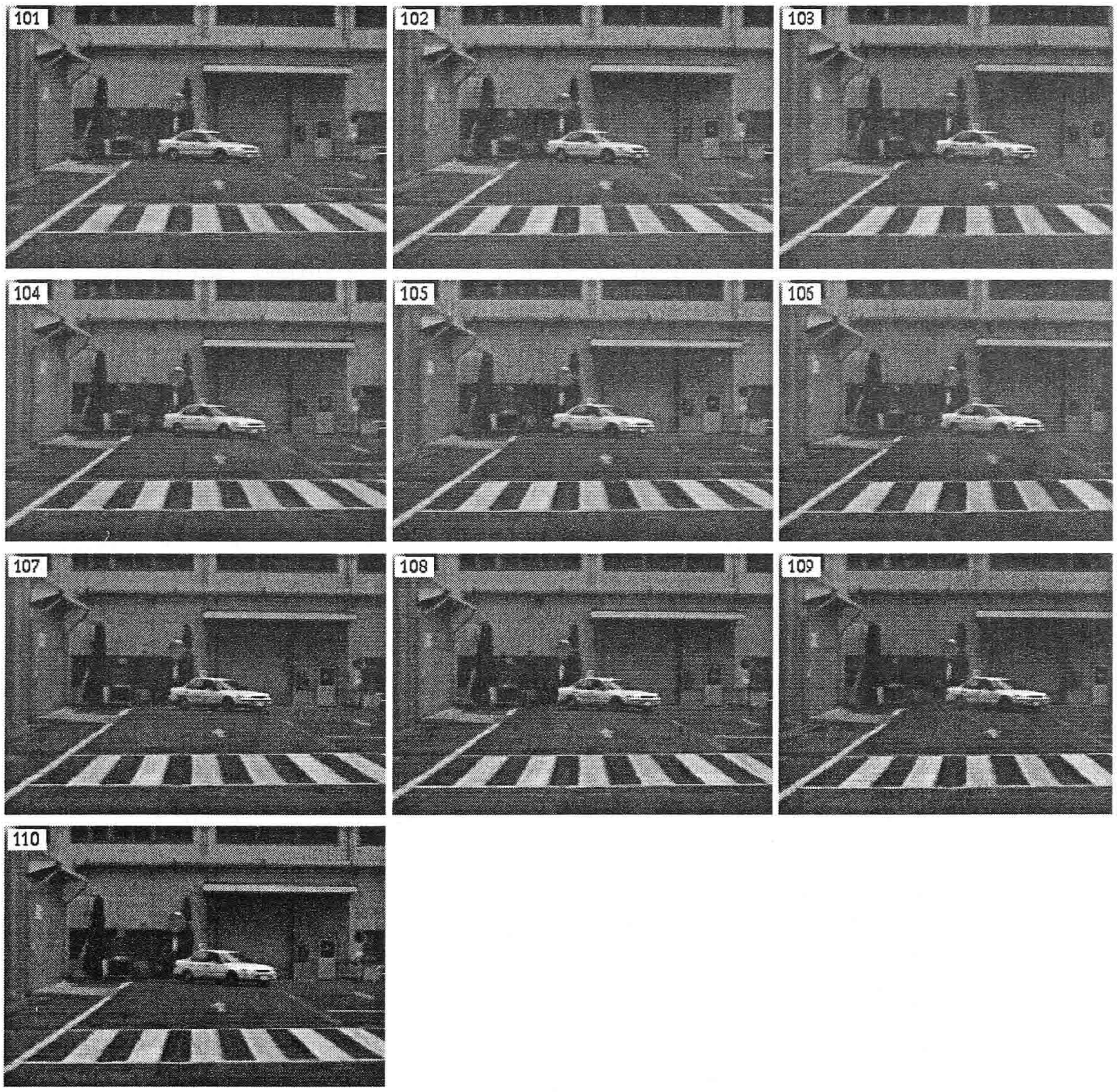


Figure 4.2: Example of optical flow detection : original image sequences. A numeral in the upper-left of each figure is a frame number.

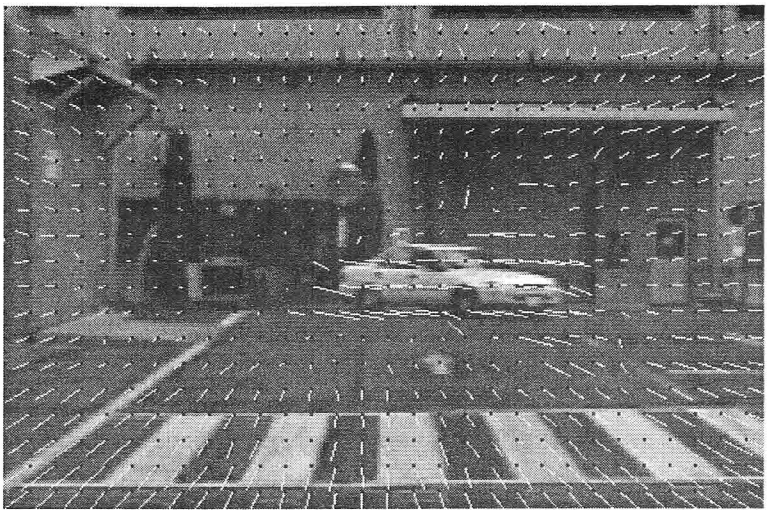


Figure 4.3: Example of optical flow detection : results. Flow vectors are shown at an interval of four pixels.

Example of optical flow detection is shown in Fig.4.2 and Fig.4.3.

In the scene, a car traverses in front, while an observer is moving straight ahead.

4.2.2 Moving obstacle candidate detection

As an observer moves through a world of static objects, the visual world projected on the retina seems to flow past. In fact, for a given direction of translatory motion and direction of gaze, the world seems to be flowing out of one particular retinal point called Focus of Expansion (FOE) as shown in Fig.4.4. Each direction of motion and gaze induces a unique FOE, which is a point at infinity if the motion is parallel to an image plane.

If an observer is moving with instantaneous velocity $(-u, -v, -w)$, keeping the coordinate system attached to the viewpoint gives points in a stationary world a relative velocity (u, v, w). Consider a point located at (x_0, y_0, z_0) at some initial time. After a time interval t, its position

in an image will be,

$$(x_t, y_t) = (\frac{x_0 + ut}{z_0 + wt}, \frac{y_0 + vt}{z_0 + wt}) \tag{4.3}$$

under the perspective projection condition.

Then, the position of the FOE is given as,

$$(x_{foe}, y_{foe}) = (\frac{u}{w}, \frac{v}{w}) \tag{4.4}$$

As shown in Fig.4.5, when there is no obstacle, every line which is an extension of an optical flow vector passes through a common intersection point which corresponds to the FOE. When an obstacle suddenly appears, on the other hand, the lines have no common intersection point. As a result, the residual error calculated in the process of estimating the FOE greatly changes, in the case when an obstacle appears, relative to the case when there is no obstacle.

Thus, the appearance of a moving obstacle is efficiently detected by using a square residual error calculated in the process of estimating the FOE[77].

If a point(x_i, y_i) has an optical flow vector (u_i, v_i), the equation of the extended flow vector is as follows:

$$a_i x + b_i y + c_i = 0 \tag{4.5}$$

Here,

$$\begin{cases} a_i = -\frac{v_i}{\sqrt{{u_i}^2 + {v_i}^2}} \\ b_i = \frac{u_i}{\sqrt{{u_i}^2 + {v_i}^2}} \\ c_i = \frac{uy_i - vx_i}{\sqrt{{u_i}^2 + {v_i}^2}} \end{cases} \tag{4.6}$$

The coefficient matrix **A** and the constant vector **c** are determined such as,

$$\mathbf{A} = \begin{bmatrix} a_1 & b_1 \\ e_2 & b_2 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix} \tag{4.7}$$

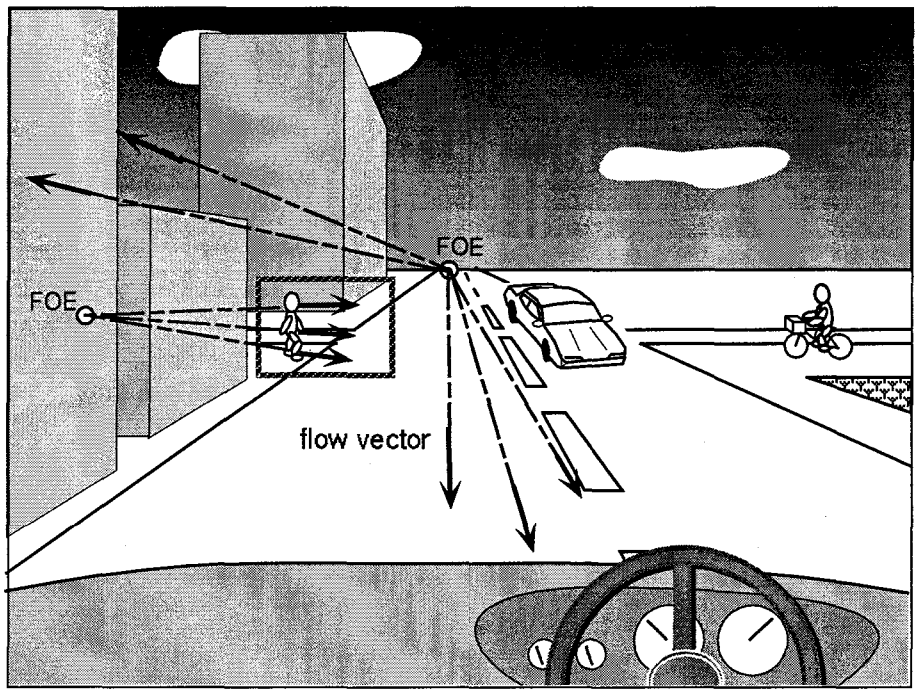


Figure 4.4: Focus of Expansion (FOE). When an observer moves through a world of static objects, the visual world projected on the retina seems to be flowing out of one particular retinal point called Focus of Expansion (FOE).

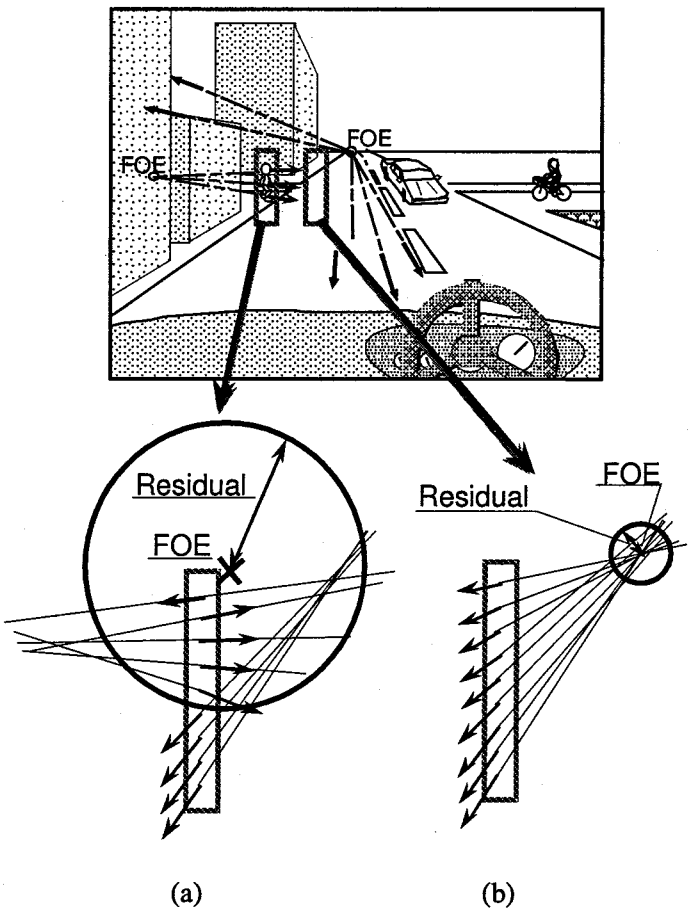


Figure 4.5: Basic idea of moving obstacle detection using FOE residual error. When there is no obstacle, every line which is an extension of the optical flow vector passes through a common intersection point which corresponds to the FOE. When an obstacle suddenly appears, on the other hand, the lines have no common intersection point. As a result, the residual error calculated in the process of estimating the FOE greatly changes, in the case when an obstacle appears relative to the case when there is no obstacle.

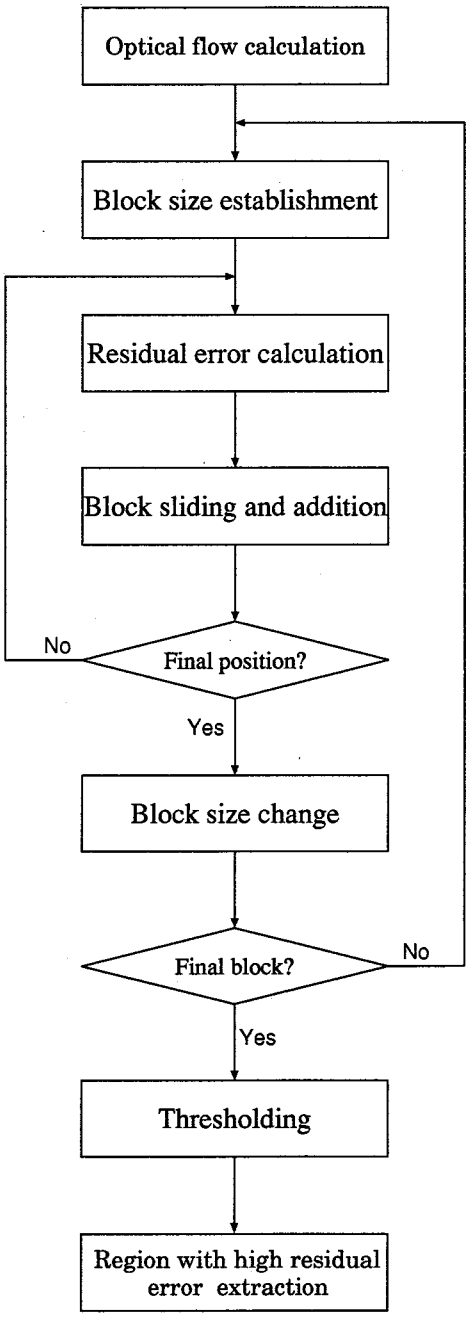


Figure 4.6: Algorithm of the moving object candidate detection process.

$$\mathbf{c} = \begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix}^T \tag{4.8}$$

Here, n is the number of flow vectors used for estimating the FOE. Then, the solution $\tilde{\mathbf{x}}$ of the following equation,

$$\mathbf{A}\mathbf{x} + \mathbf{c} = \mathbf{0} \tag{4.9}$$

which is given as,

$$\tilde{\mathbf{x}} = -\mathbf{A}^{+}\mathbf{c} \tag{4.10}$$

is the estimated FOE

Here, $\mathbf{A}^{+}$ is the pseudo inverse matrix of $\mathbf{A}$.

Then, a square residual error E is calculated as follows:

$$E = \|\mathbf{A}\tilde{\mathbf{x}} + \mathbf{c}\|^2 \tag{4.11}$$

There are three main possibilities when the square residual error for a region containing a moving obstacle is falsely small instead of large:

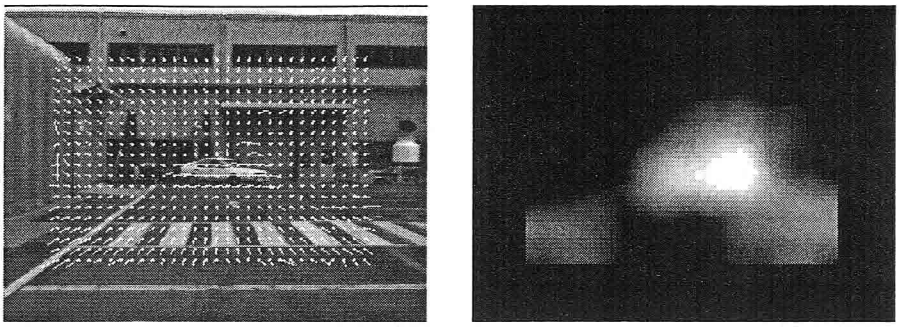
(1) The block contains almost all flow vectors caused by the motion of an obstacle.

(2) The boundary of a block touches the boundary of an obstacle.

(3) A size of an obstacle is smaller than the size of a block.

To detect the obstacle in the three cases stably, rectangular sliding-blocks of multiple sizes are used. The reason for using rectangular blocks is just simplicity. When the background region is sufficiently larger than each region of moving obstacles, appropriate rectangular blocks which involve both a background region and an obstacle region, and covered the obstacle region, can be set in the image. The block sizes are experimentally determined according to the sizes, the minimal distances to detect and velocity of moving objects.

A square residual error is first calculated in a block by using (4.11) and normalized by the number of flow vector. Then, the normalized square residual error value is additionally voted at corresponding points of the block in a memory (we call it a **residual image**), which has the same size as an image. This process is repeatedly executed with sliding the block and changing the size of the block. Values in overlapped regions among multiple blocks are averaged and voted.



(a) Optical flow field (b) Residual image

Figure 4.7: Example of a residual image. Brightness of the residual image is proportional to an amount of the square residual error values. In (b) the region of an obstacle(car) has a high residual error value in comparison with the background region.

An example of moving obstacle candidate detection is shown in Fig.4.7, where Fig.4.7(a) shows an optical flow field extracted by using the Weber-Malik method shown in the last section, and the residual image is in Fig.4.7(b).

Brightness of the image is proportional to an amount of the square residual error values. This figure clearly shows that the region of the obstacle has a high residual error value in comparison with the background region.

In order to suppress noise in the residual image caused by erroneous flow vectors, a sequence of residual images is convoluted with a temporal low-pass filter. Finally, regions which have larger values than a threshold value are extracted as obstacle candidate regions. The threshold value is determined experimentally.

4.2.3 Moving object recognition

We propose a simple moving object recognition method which analyzes optical flow information.

In this method, the moving object recognition problem is formulated as the analysis of differences in local motion that are inherent to the categories of an object to be recognized. We use the following practical assumption: *parts with independent motion are projected to regions which have separated patterns in optical flow field.* As a result, the distribution patterns in some feature space show inherent aspects, that are useful for recognition.

The distribution pattern in the 2D feature (optical flow direction, intensity) space is schematically shown in Fig.4.8.

When the independent moving part of an object is one, such as a car, the optical flow directions are similar for components, such as a body frame, window glasses or wheels, because all components move together. As a result, a distribution pattern only expands in the intensity direction. On the other hand, the number of independent moving parts, such as humans or bicycles, increases and the distribution pattern expands in the intensity direction and the optical flow direction.

The observed distribution patterns involved in obstacle candidate detection results are not separated as in Fig.4.8, but blurred (connected) as in Fig.4.9. The reasons are as follows:

- The distribution pattern blurs in the intensity direction, because a result of optical flow estimation also blurs due to using spatio-temporal filters, and the detection units are rectangular blocks which also involve background regions.

- The distribution pattern blurs in the optical flow direction due to errors in estimating optical flow.

- False patterns due to failure of obstacle candidates detection are mixed in the feature space.

To extract the difference of distribution characteristics in the 2D feature space in spite of these problems, we used the distribution width values around the principal axes of inertia and the direction of the

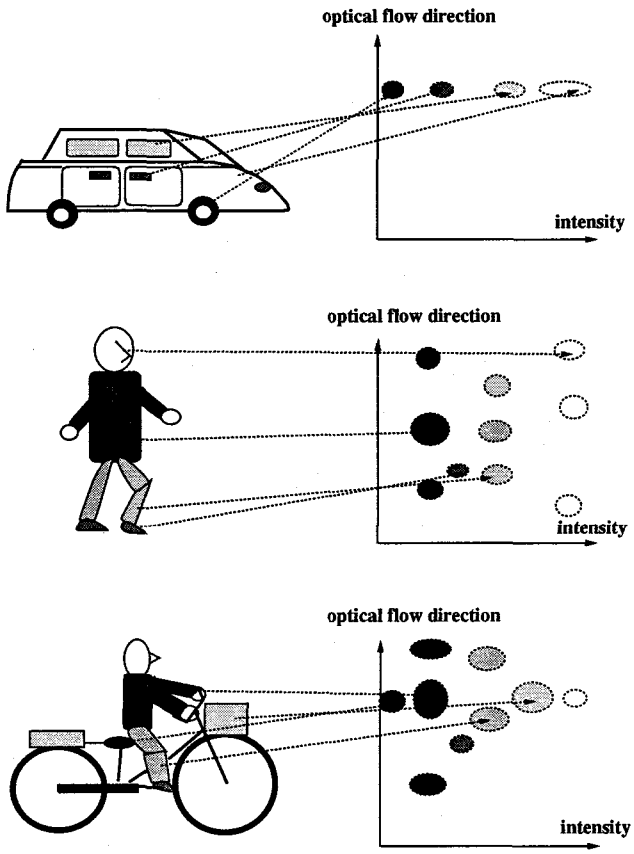


Figure 4.8: Schematic distribution pattern in flow direction-intensity space. Color of an ellipse in a distribution pattern graph at right side of the figure corresponds to the intensity of an element in an object.

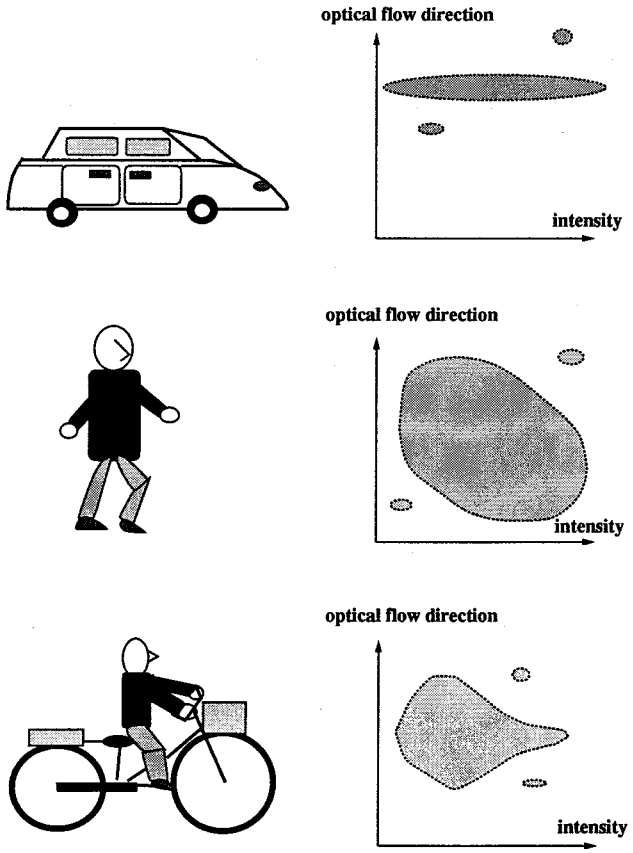


Figure 4.9: Observed distribution pattern in flow direction-intensity space. The real observed distribution patterns are not separated as in Fig.4.8 but connected.

principal axes in the distribution pattern that are calculated by the principal component analysis.

The moving object recognition process is shown in Fig.4.10.

First, the directions of optical flow and intensity values are stored for the pixels involved in each candidate region during a few frames just after moving object candidates are initially detected. Using the stored data, the proportion (coefficient of determination) and the direction values of the first principal components are calculated by using the principal component analysis.

The general process of principal component analysis is shown below.

From the n data $\{x_{ij}\}$, $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, n$, the covariance matrix $\mathbf{S}$ is calculated as follows:

$$\mathbf{S} = \begin{bmatrix} {s_1}^2 & s_{12} & \cdots & s_{1p} \\ s_{12} & {s_{22}}^2 & \cdots & s_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1p} & s_{2p} & \cdots & {s_p}^2 \end{bmatrix} \tag{4.12}$$

Next, the eigenvalue λ of $\mathbf{S}$ is estimated by solving the following characteristic equation.

$$det \begin{vmatrix} {s_1}^2 - \lambda & s_{12} & \cdots & s_{1p} \\ s_{12} & {s_{22}}^2 - \lambda & \cdots & s_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1p} & s_{2p} & \cdots & {s_p}^2 - \lambda \end{vmatrix} = 0 \tag{4.13}$$

The solution λ_i of (4.13) is all positive definite, that is,

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0 \tag{4.14}$$

Finally, an eigenvector $\mathbf{a_i} = (a_{i1}, a_{i2}, \cdots, a_{ip})$ which corresponds to each eigenvalue λ_i is calculated by solving (4.15).

$$\mathbf{S}\mathbf{a_i} = \lambda_i \mathbf{a_i} \tag{4.15}$$

Here,

$${a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{ip}}^2 = 1 \tag{4.16}$$

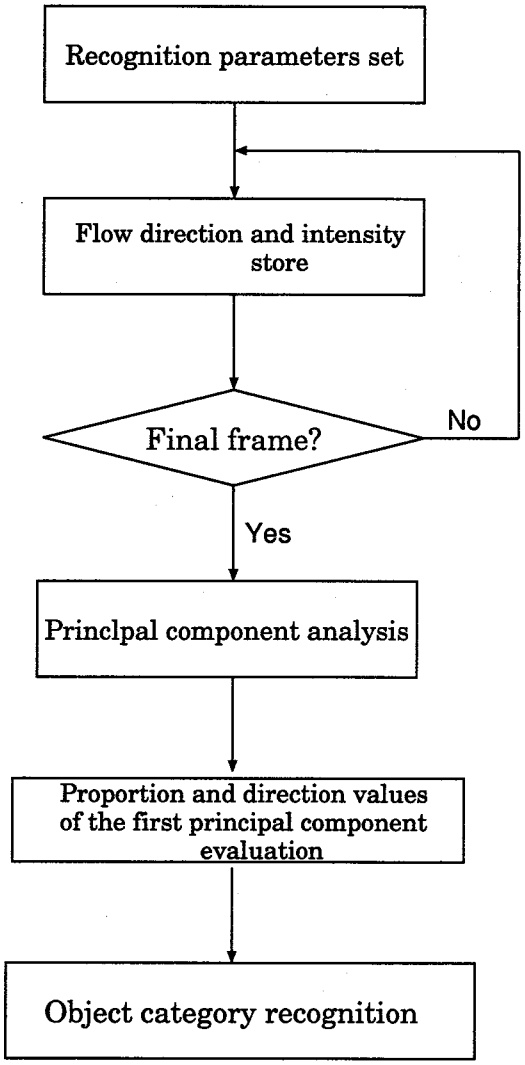


Figure 4.10: Algorithm of the moving object recognition process.

As a result, the i-th principal component Z_i can be represented as follows:

$$Z_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ip}x_p \tag{4.17}$$

The proportion value p_i of the i-th principal component Z_i represents the ratio of original information described by the principal component.

$$p_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_p} \tag{4.18}$$

By applying the above analysis for two-dimensional data (n = 2), the local motion information of multiple points involved in moving object candidate regions can be efficiently described by the first dominant component as shown in Fig.4.11.

Even if parts of background are mixed in the candidate regions, the results of the principal component analysis are not be affected so much.

In the MOROFA method, each candidate is classified into a category of object that is expected to appear in the scene, by comparing the following two kinds of values with standard data ranges, that is, the direction values of the eigenvectors which correspond to the first eigenvalues $\mathbf{d}(\mathbf{a_1})$ and the proportion values $\mathbf{p_1}$ given by (4.18).

$$\mathbf{p_1} = \frac{\lambda_1}{\lambda_1 + \lambda_2} \tag{4.19}$$

In MOROFA method, objects to be recognized in outdoor environments are classified into four categories; *cars, bicycles, pedestrians* and *unknown objects.*

When the independent moving part of an object is one, the first principal component represents the unique motion which coincides with the moving direction of the object itself and the proportion value $\mathbf{p_1}$, which indicates the inverse of the variance, is nearly 1.0. The direction of the first principal component $\mathbf{d}(\mathbf{a_1})$ is nearly 0.0 because most points involved in the moving object candidate region have similar optical flow directions.

While the number of independent moving parts increases, the proportion value $\mathbf{p_1}$ decreases from 1.0 because several optical flow patterns

are mixed and the variance around the principal axis in the distribution pattern becomes larger. In addition, the direction of the first principal component $\mathbf{d}(\mathbf{a}_1)$ also increases. Consequently, the recognition of moving objects can be efficiently realized by analyzing the proportion value and the direction of the first principal component as shown in Fig.4.11.

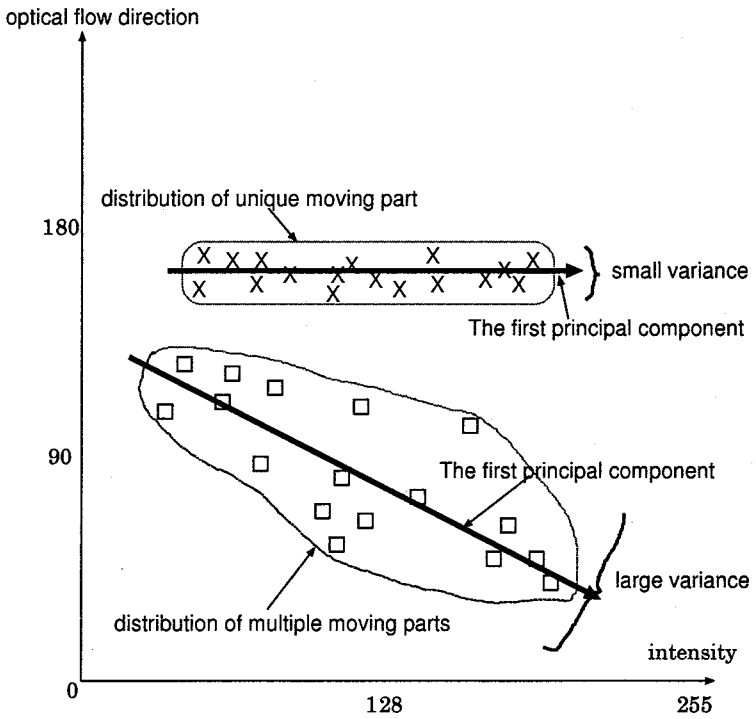


Figure 4.11: The principal components of distribution patterns in the feature space.

The recognition is executed by checking if both the values ($\mathbf{p}_1$ and $\mathbf{d}(\mathbf{a}_1)$) are involved in some predetermined cluster as shown in Fig.4.12. Distribution patterns of cars have larger ($\mathbf{p}_1$ values and smaller $\mathbf{d}(\mathbf{a}_1)$ values, on the other hand, those of humans have smaller $\mathbf{p}_1$ values and larger $\mathbf{d}(\mathbf{a}_1)$ values. Those of bicycles are located in the middle. A distribution pattern in other regions is judged as a pattern of an unknown object.

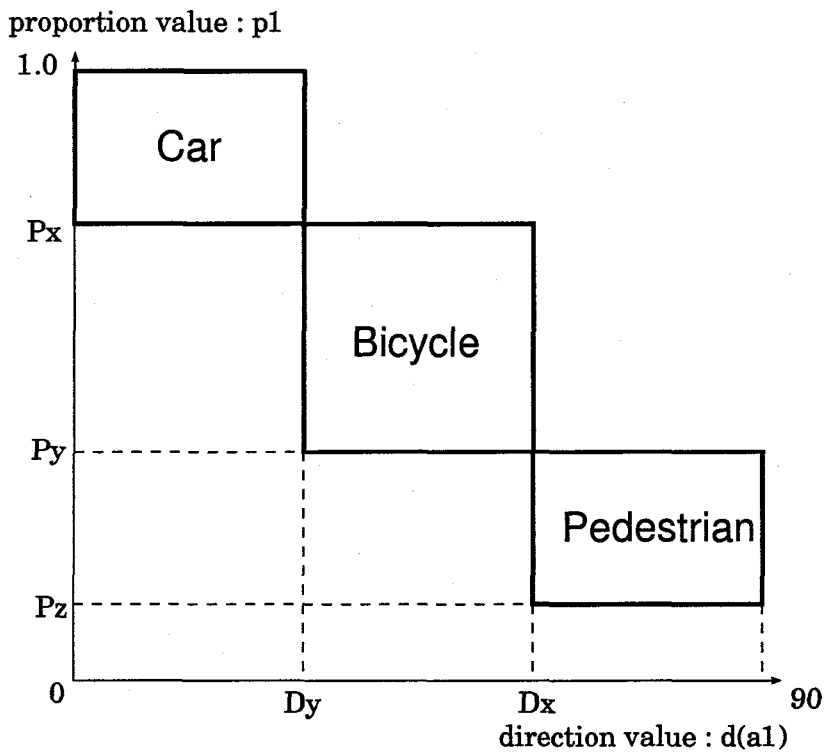


Figure 4.12: The clusters for recognition. Parameters Px, Py, Pz, Dx and Dy are determined by preliminary experiments.

The MOROFA method is stable against errors occurring in optical flow fields and in the detection of moving object candidates because the method is based on the statistical analysis and does not use shape information for recognition as do conventional recognition methods. In addition, the MOROFA method can recognize a moving object efficiently, because no apparent motion segmentation process is necessary such as in [57]. Owing to the small number of the parameters requiring adjustment, this method can be applied to many industrial fields (for example, an intelligent machine surveillance system or an obstacle detection system for an autonomous vehicle).

4.3 Experimental results

In this section, experimental results using real images in two factories of Toshiba Corp. (Osaka, Fuchu) are presented. Images for the experiments were taken from a camera attached to the roof of an automobile which was driven manually at the speed of 10 Km/h.

The number of independent local moving part is one in the case of cars, three in the case of bicycles (the bicycle, the leg of the rider and the hand of the rider) and five in the case of pedestrians (the upper half of a body, the left hand, the right hand, the left leg and the right leg).

At first, the range parameters for the distribution width values around the principal axes of inertia and the direction values of the principal axes shown in Fig.4.12 were experimentally determined using the data of five scenes (scene 1, 2, 4, 6 and 7) in the Osaka factory as shown in Table tbl:obst-param. The length of the data storage period was determined as 10 frames. The same parameter values were available for the remaining scenes.

For moving obstacle candidate detection, we used three sizes of sliding blocks as shown in Table.4.1, that were determined experimentally.

The configuration of the scenes is shown in Table 4.3. Scene 1 $\sim$ 14 were taken in the Osaka factory and scene 15 $\sim$ 25 were taken in the Fuchu factory.

Table 4.1: Sliding-block configuration.

	size	sub-sampling interval	sliding interval
block 1	32×32	4×4	4
block 2	64×64	8×8	4
block 3	96×96	12×12	4

Table 4.2: Values of parameters for recognition.

proportion parameters			**direction parameters (degree)**	
Px	**Py**	**Pz**	**Dx**	**Dy**
0.75	**0.60**	**0.50**	**30**	**5**

Examples of distribution pattern in flow direction-intensity space are shown in Fig.4.13 $\sim$ Fig.4.15.

Those patterns qualitatively coincide with the schematic patterns shown in Fig.4.9.

Examples of detecting moving object candidates (nine successive frames) are shown in Fig.4.16, Fig.4.17 and Fig.4.18. The shadowed regions are detection results of moving object candidates.

Though detection of moving object candidates is generally correct in scenes 1 $\sim$ 8, several errors such as missing of objects or detection of a part of road occur in scenes 9 $\sim$ 25.

The result of the recognition is shown in Table 4.4.

The average rate of the successful recognition was 84%, and the average rate of the recognition failure was 8%.

As to those scenes whose results of moving object candidates detection are generally correct (scenes 1 $\sim$ 8), the rate of the successful recognition was 100%.

Table 4.3: Scene configuration.

	moving object category	moving direction
scene 1	Car	horizontal crossing (from right to left)
scene 2	Car	horizontal crossing (from left to right)
scene 3	Car	turning to the right
scene 4	Bicycle	horizontal crossing (from left to right)
scene 5	Bicycle	oblique crossing (from left deep to right shallow)
scene 6	Pedestrian	approaching (left side of road)
scene 7	Pedestrian	horizontal crossing (from left to right)
scene 8	Pedestrian	horizontal crossing (from right to left)
scene 9	Car	preceding
scene 10	Car	turning to the left
scene 11	Car	turning to the right
scene 12	Car	preceding
scene 13	Bicycle	approaching (left side of road)
scene 14	Bicycle	oblique crossing (from left shallow to right deep)
scene 15	Bicycle	horizontal crossing (from right to left)
scene 16	Bicycle	oblique crossing (from right shallow to left deep)
scene 17	Bicycle	going away (left side of road)
scene 18	Bicycle	oblique crossing (from right deep to left shallow)
scene 19	Pedestrian	going away
scene 20	Pedestrian	oblique crossing (from left deep to right shallow)
scene 21	Pedestrian	oblique crossing (from left shallow to right deep)
scene 22	Pedestrian	horizontal crossing (from left to right)
scene 23	Pedestrian	oblique crossing (from left shallow to right deep)
scene 24	Pedestrian	approaching (left side of road)
scene 25	Pedestrian	going away (left side of road)

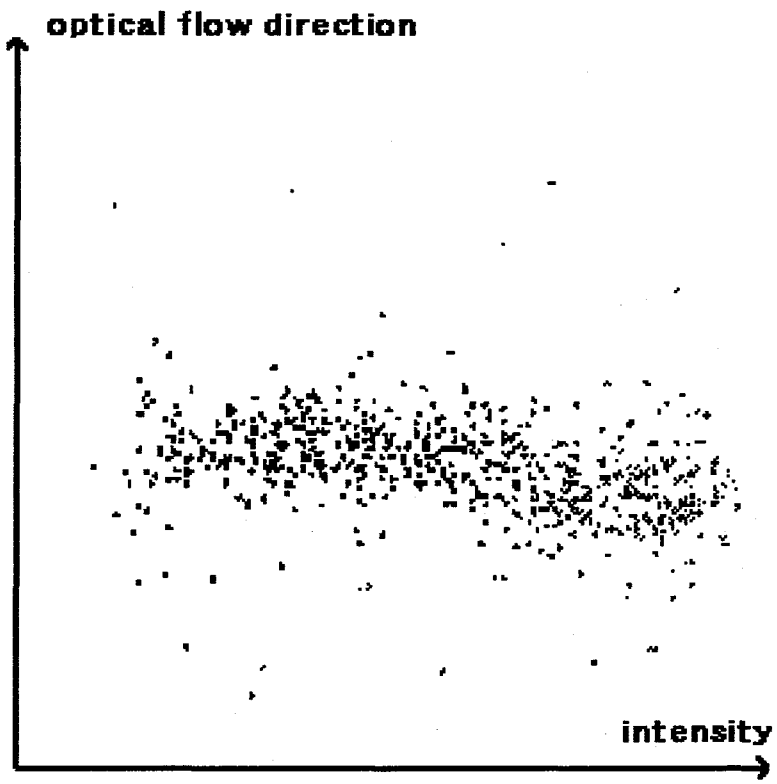


Figure 4.13: Example of distribution pattern (car, turning to the right in scene 3).

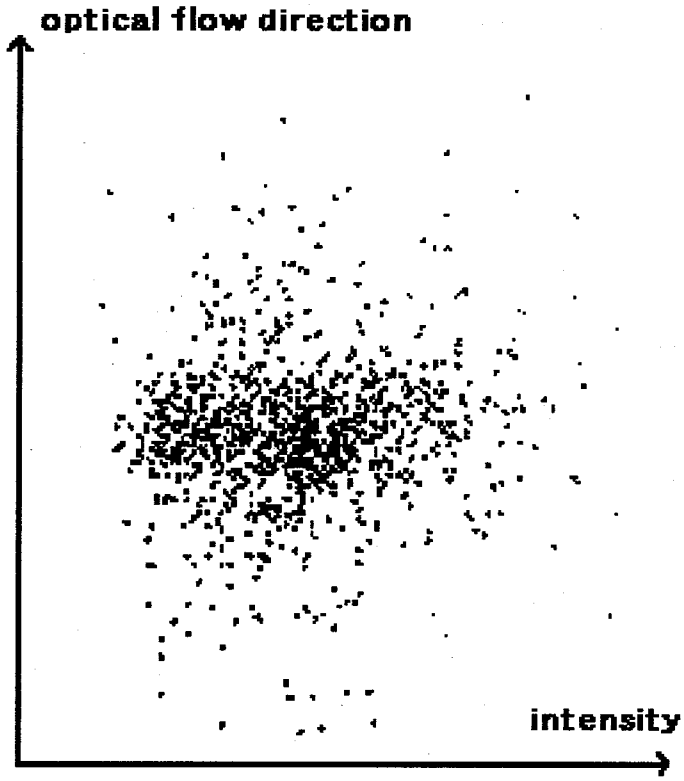


Figure 4.14: Example of distribution pattern (bicycle, oblique crossing in scene 5).

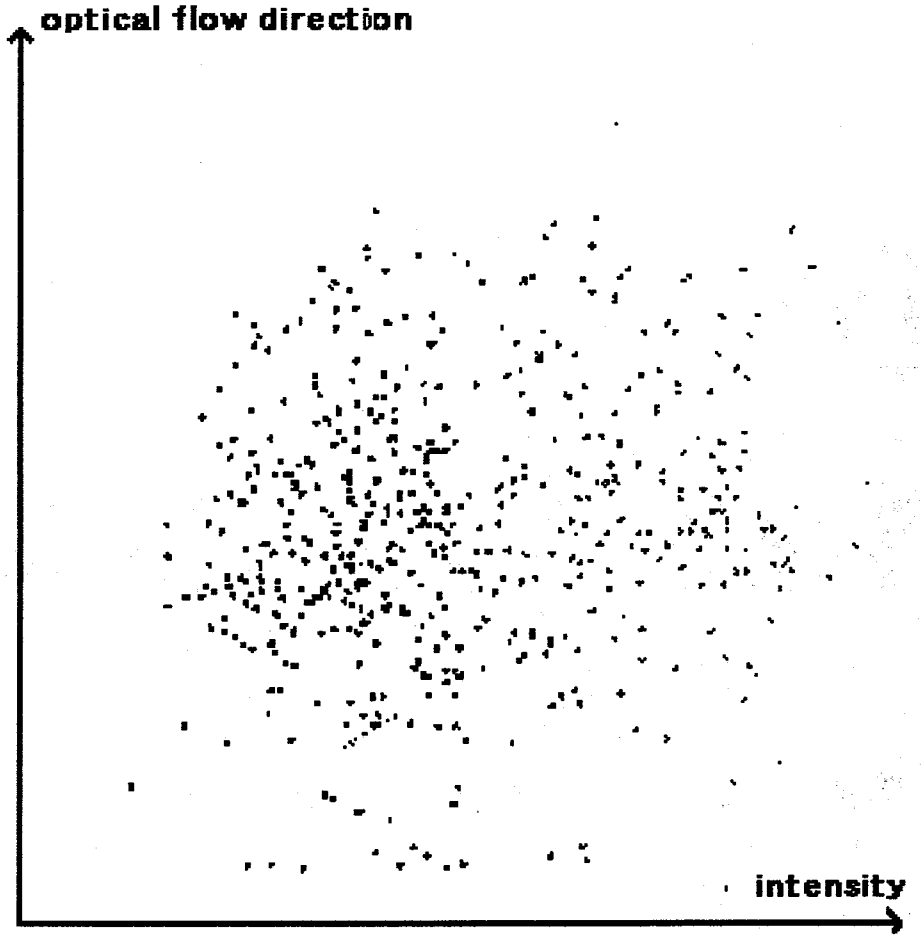
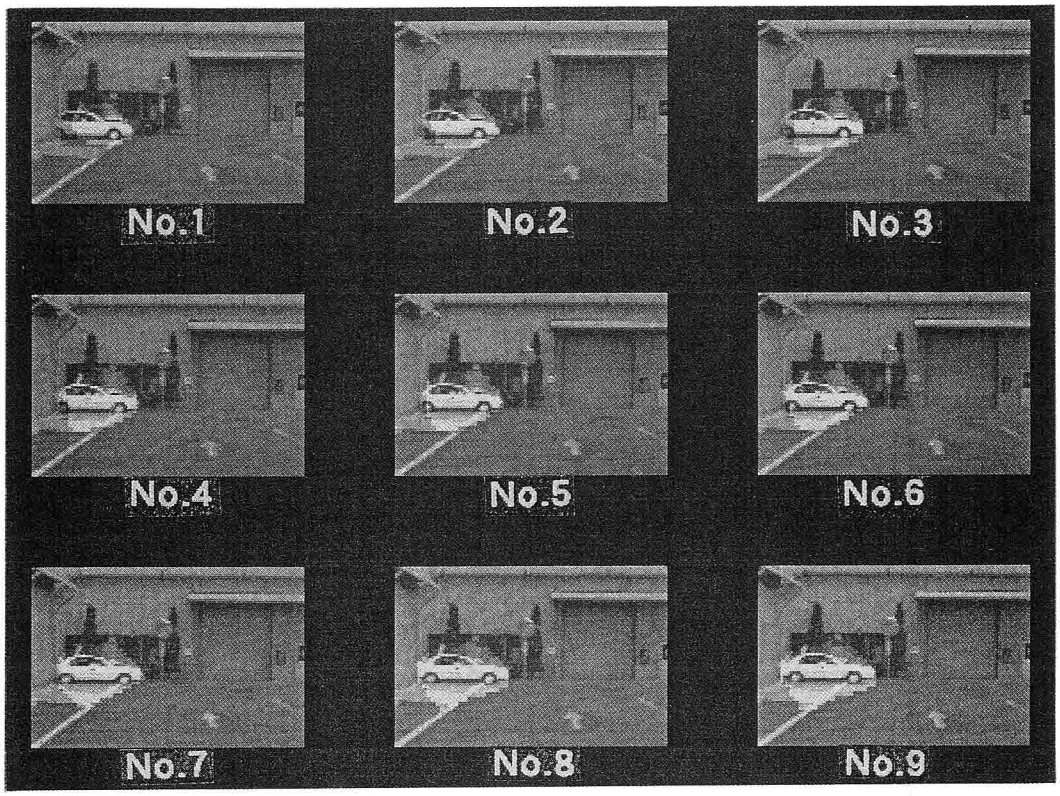
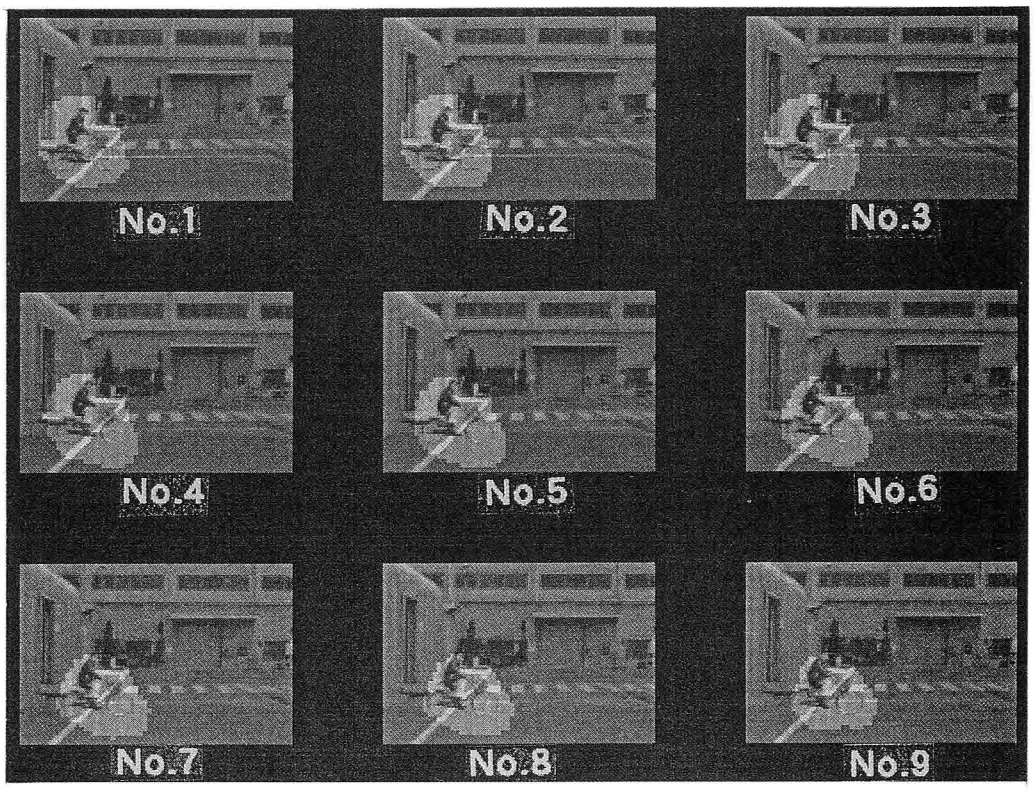


Figure 4.15: Example of distribution pattern (pedestrian, horizontal crossing in scene 7).



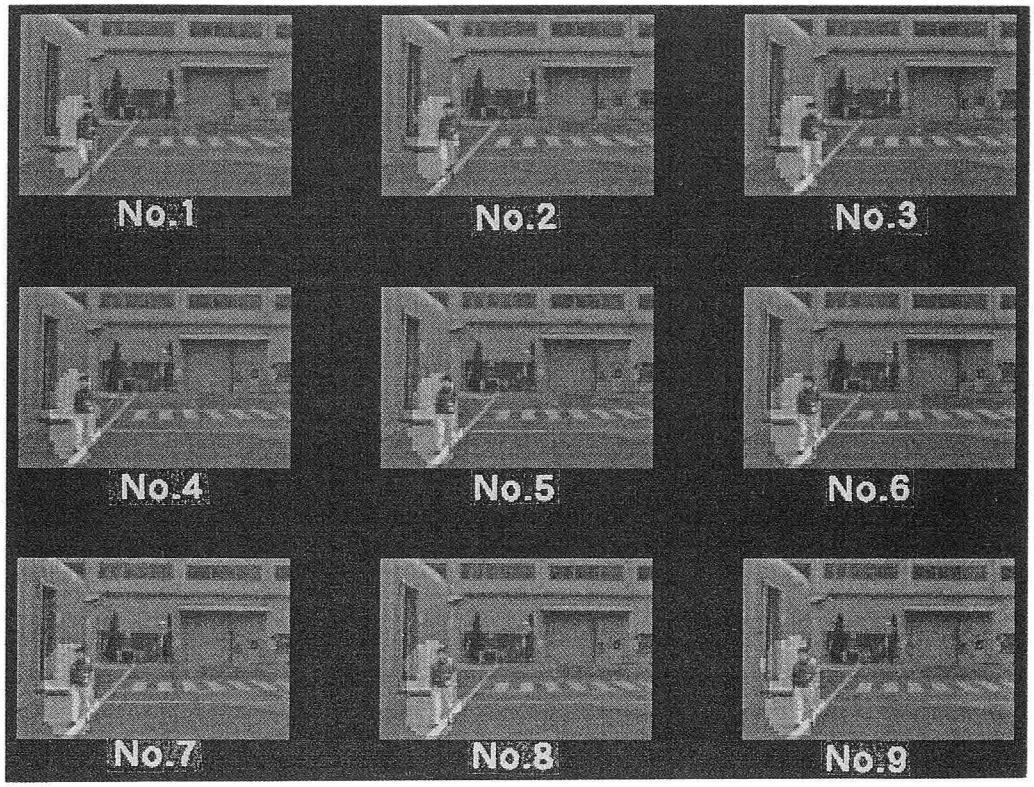
The bright regions are results of moving object candidate detection.

Figure 4.16: Moving object candidates detection results (car in scene 2).



The bright regions are results of moving object candidate detection.

Figure 4.17: Moving object candidates detection results (bicycle in scene 4).



The bright regions are results of moving object candidate detection.

Figure 4.18: Moving object candidates detection results (pedestrian in scene 6).

Table 4.4: Recognition results of the MOROFA method.

	moving object category	recognition result
scene 1	Car	Car
scene 2	Car	Car
scene 3	Car	Car
scene 4	Bicycle	Bicycle
scene 5	Bicycle	Bicycle
scene 6	Pedestrian	Pedestrian
scene 7	Pedestrian	Pedestrian
scene 8	Pedestrian	Pedestrian
scene 9	Car	Car
scene 10	Car	Car
scene 11	Car	Car
scene 12	Car	Car
scene 13	Bicycle	Pedestrian
scene 14	Bicycle	unknown object
scene 15	Bicycle	Pedestrian
scene 16	Bicycle	Bicycle
scene 17	Bicycle	Bicycle
scene 18	Bicycle	Bicycle
scene 19	Pedestrian	Pedestrian
scene 20	Pedestrian	Pedestrian
scene 21	Pedestrian	unknown object
scene 22	Pedestrian	Pedestrian
scene 23	Pedestrian	Pedestrian
scene 24	Pedestrian	Pedestrian
scene 25	Pedestrian	Pedestrian

The processing time was about 29 seconds when a test program written in C language was used on a standard engineering workstation; the time for optical flow calculation was 13 seconds per frame, moving object candidate detection was 15 seconds per frame and moving object recognition was less than 1 second.

4.3.1 Discussions

One of the main reasons for recognition failure is the coincidence of the moving directions of the moving observer and the moving object (scene 13). That is, the detection of a moving object becomes difficult when the FOE positions of a moving observer and that of a moving object approach each other. To overcome this problem, we have been improving the object candidate detection process by using directional information from the optical flow field in addition to the FOE position. Even in such difficult cases, the recognition could be performed correctly when the major part of moving object regions were detected (scenes 6, 9, 12, 17, 19, 24 and 25).

Another reason for recognition failure is the partial deficit in the detected moving object candidate region. In the case of scenes 14, 15 and 21, the recognition failed because not enough local motion information for recognition could be acquired as a result of the missing of objects and detection failure for a part of road. To counter this problem and improve the recognition ability, we have been investigating the use of inherent shape information (for example, circles of wheels in the case of bicycles and cars), for verification in addition to the MOROFA method. If the number of objects to be recognized increases, the addition of features should be also investigated.

There are several problems in applying this method to real applications. The main problems and solution plans are as follows:

- *Camera vibration caused by motion oscillation of observer:*

  Because the effect of camera vibration influences the whole image, the obstacle candidates detection process using a square residual error is not influenced by the translational motion component of camera vibration. To eliminate the rotational motion component of camera vibration, flow vector subtraction at far background,

such as cloud region in the sky, where the translational component of optical flow vectors becomes nearly 0, is possible.

- *Existence of multiple obstacles:*

  Generally, the regions of obstacle candidates are separated when they appear. In that case, the kinds of obstacles are successfully recognized by using optical flow information involved in each candidate region independently.

- *Existence of shadows:*

  The optical flow vectors due to shadow of moving observer disturb the obstacle candidate detection and recognition process. In that case, the optical flow vector involved in shadow candidate region, which is detected by using intensity information or height information (shadow is projected on the road plane), is eliminated before the obstacle candidate detection process.

- *Correspondence of moving direction between observer and obstacles:*

  When the moving direction of an observer and an obstacle becomes similar, that is, an obstacle approaches from front direction, the obstacle candidate detection becomes difficult because FOE position of background and that of the obstacle approach each other. Even in such a difficult case, the optical flow of the obstacle can be discriminated from that of background region by using the length information of optical flow vectors, because the obstacle is much nearer than background region and has long flow vectors.

- *Speed up*

  When an observer is moving at 20 Km/h (normal speed in a factory) and an obstacle traverses 50m in front of the observer, the obstacle detection and recognition process has to be finished within 8 seconds.

  We have improved the optical flow calculation process and the obstacle detection candidate detection process. The time required

for optical flow calculation is 4.4 seconds per frame and the time required for obstacle candidate detection is 3.4 seconds per frame. We intend to develop hardware to detect the optical flow field in real-time.

Chapter 5

Conclusion

The subject of this thesis is research and development conducted with a view to realizing an autonomous mobile robot able to work in an unstructured dynamic environment such as an office, a street or a home. Specially, this thesis describes an architecture of a controller and vision-based obstacle detection and recognition methods.

To attain both efficient navigation to a final goal and robustness against various kinds of failures that may occur in a dynamic environment, a task-oriented three-grouped control architecture, which consists of Reflexive Behavior Agent Group for self-continuance, Purposive Navigation Behavior Agent Group and Adaptive Behavior Agent Group, has been proposed.

The Reflexive Behavior Agent Group, consists of agents with contact, infrared and ultrasonic ranging sensors, which maintain the minimal safety of the robot and humans at least.

The role of the Purposive Navigation Behavior Agent Group is to efficiently navigate the robot to a final goal. Several target searching and tracking agents in accordance with subgoals, such as a wall or an intersection, have been installed.

The Adaptive Behavior Agent Group, consisting of Free-Space-Explorer, Obstacle-Boundary-Follower, Open-Space-Explorer and Wall-Follower, has been developed to recover from failures in the Purposive Navigation Behavior agents or infinite-loop motion sequence situations.

An extended energy-minimizing method is developed to produce an appropriate velocity and steering commands for the robot navigation

using multiple sensor readings.

These grouped agents are directly connected to a navigation actuator controller named Motion Executor, which determines the motion (velocity and steering) command of the robot. Independent agent-arbitration means are prepared in the Motion Executor.

In addition to conducting simulation experiments to prove the effectiveness of the proposed architecture, we have developed a small cart-type robot **BIRDIE** and have done several indoor navigation experiments, such as moving obstacle (human) avoidance, intersection tracking and total navigation in an office, successfully.

As for the attainment of the vision-based obstacle detection and recognition abilities, two kinds of methods have been proposed.

At first, to realize the vision-based free space detection function, which is important for attaining safe navigation, the Disparity Prediction Stereopsis (DPS) method using binocular stereo cameras has been proposed for static obstacle detection.

The DPS method consists of four parts: the feature edge extraction part, the vertical edge matching part, the obstacle prediction region establishment part and the collision-free space detection part. The existence of obstacles in the space where a robot is due to move can be efficiently examined without making a depth map for the whole scene.

Experimental results of detecting several static obstacles in indoor environments and the carrying out, more than 100 times, of real navigation experiments using a clover-type vehicle AIMARS have demonstrated the effectiveness of the DPS method.

Next, a vision-based moving obstacle detection and recognition method which is important for attaining both safety and flexibility in navigation has been described.

To realize these requirements, the MOROFA method which analyzes optical flow information has been proposed.

The MOROFA method consists of three parts: the optical flow calculation part, the moving object candidate detection part and the moving object recognition part. The candidates of moving obstacles can be effectively detected by calculating the square residual errors in the process of estimating the FOE in local windows, without estimating the motion parameters which would impose a large computational cost and it is not robust against noise. In addition, the types of the objects can

be robustly recognized by analyzing the optical flow information in 2D feature space.

Experimental results of moving obstacle (cars, bicycles, and pedestrians) recognition in two factories have demonstrated the effectiveness of the proposed method.

The remaining problems include attainment of environment learning ability for adaptation and reactive planning ability for flexible driving command determination.

The author believes that, in the 21st century, practical intelligent mobile robots will be working cooperatively with humans in several domains, for example, as guides for aged persons or as personal assistants in offices, as a result of the combination of three strands of robotics research, namely pursuit of navigation performance, acquisition of adaptability and realization of reactive robustness.

Bibliography

[1] R.A.Brooks: "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, Vol.2, No.1, pp.14-23, 1986.

[2] S.Tsuji: "Monitoring of a building environment by a mobile robot", in Proc. of International Symposium on Robotics Research (ISRR'84), pp.279-286, 1984.

[3] M.Asada: "Building a 3-D world model for a mobile robot sensory data", in Proc. of the 1988 International Conference on Robotics and Automation (ICRA'88), pp.918-923, 1988.

[4] H.Ishiguro, M.Yamamoto and S.Tsuji : "Omni-directional Stereo for Making Global Map", in Proc. of the Third International Conference on Computer Vision (ICCV)'91, pp.540-547, 1991.

[5] K.Yamazawa, Y.Yagi and M.Yachida: "Obstacle Detection with Omnidirectional Image Sensor HyperOmni Vision", in Proc. of the 1988 International Conference on Robotics and Automation (ICRA'95), pp.1062-1067, 1995.

[6] M.Asada, S.Noda, S.Tawaratsumida and K.Hosoda: "Vision-Based Reinforcement Learning for Purposive Behavior Acquisition", in Proc. of the 1995 International Conference on Robotics and Automation (ICRA'95), pp.146-153, 1995.

[7] J.L.Jones and A.M.Flynn: "Mobile Robots - Inspiration to Implementation", A K Peters, Ltd., ISBN L-56882-0LL-3, 1993.

[8] S.Tsugawa, T.Yatabe, T.Hirose and S.Matsumoto: "An automobile with artificial intelligence", in Proc. of the 6th International Joint Conference on Artificial Intelligence (IJCAI-79), pp.893-895, 1979.

[9] V.Graefe: "Dynamic vision systems for autonomous mobile robot", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'89, pp.12-23, 1989.

[10] K.Nishikawa and H.Mori: "Rotation Control and Motion Estimation of Camera for Road Following", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'93, pp.1313-1318, 1993.

[11] H.Mori, K.Nishikawa and S.Kotani: "A Locomotion Performance Learning of the Mobile Robot", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'95, pp.447-452, 1995.

[12] G.Giralt, R.Sobek and R.Chatila: "A Multi-Level Planning and Navigation System for a Mobile Robot; A First Approach to HILARE", in Proc. of the 6th International Joint Conference on Artificial Intelligence (IJCAI-79), pp.335-337, 1979.

[13] M.Ohzora, T.Ozaki, S.Sasaki, M.Yoshida and Y.Hiratsuka: "Video-Rate Image Processing System for an Autonomous Personal Vehicle System", in Proc. of IAPR Workshop on Machine Vision and Applications(MVA)'90, pp.389-392, 1990.

[14] C.E.Thorpe(edited): "**Vision and Navigation** The Carnegie Mellon Navlab", Kluwer Academic Publishers, ISBN 0-7923-9068-7, 1990.

[15] C.E.Thorpe, M.H.Herbert and T.Kanade: "Vision and Navigation for the Carnegie-Mellon Navlab", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 10(3), pp.362-373, 1988.

[16] H.P.Moravec: "The Stanford Cart and the CMU Rover", in Proc. of the IEEE, Vol.71, No.7, pp.872-884, 1983.

[17] K.Onoguchi, M.Watanabe, Y.Okamoto and H.Asada: "Visual Navigation System for a Mobile Robot", in Proc. of International Workshop on Intelligent Robots and Systems (IROS)'89, pp.590-597, 1989.

[18] K.Onoguchi, M.Watanabe, Y.Okamoto, Y.Kuno and H.Asada: "A Visual Navigation System using a Multi-Information Local Map", in Proc. of the 1990 International Conference on Robotics and Automation (ICRA'90), pp.767-774, 1990.

[19] K.Onoguchi, M.Watanabe, Y.Okamoto and Y.Kuno: "Multi-Information Local Map for Visual Navigation", Journal of the Robotics Society of Japan (JRSJ), Vol.11, No.3, pp.401-409, 1993 (in Japanese).

[20] S.Yuta, S.Suzuki and S.Iida: "Implementation of a small size experimental self-contained autonomous robot sensors, vehicle control, and description of sensor based behavior", in Proc. of 1993 International Symposium On Experimental Robotics (ISER), pp.344-359, Springer-Verlag, 1993.

[21] K.Nagatani and S.Yuta: "Path and Sensing Point Planning for Mobile Robot Navigation to Minimize the Risks of Collision", Journal of the Robotics Society of Japan (JRSJ), Vol.15, No.2, pp.197-206, 1997 (in Japanese).

[22] A.M.Waxman, J.J.Lemoigne, L.S.Davis, B.Srinivasan, T. R.Kushner, E.Liang and T.Siddalingaiah: "A Visual Navigation System for Autonomous Land Vehicles", IEEE Journal on Robotics and Automation, 1987.

[23] M.A.Turk, D.G.Morgenthaler, K.D.Gremban and M.Marra: "VITS - A Vision System for Autonomous Land Vehicle Navigation", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 10(3), pp.342-361, 1988.

[24] T.Shibata, Y.Matsumoto, T.Kuwahara, M.Inaba and H.Inoue: "Hyper Scooter: a Mobile Robot Sharing Visual Information with

a Human", in Proc. of the 1995 International Conference on Robotics and Automation (ICRA'95), pp.1074-1079, 1995.

[25] R.Volpe, J.B.Balaram, T.Ohm and R.Ivlev: "The Rocky 7 Mars Rover Prototype", International Workshop on Intelligent Robots and Systems(IROS)'96, pp.1558-1564, 1996.

[26] H.Inoue, T.Tachikawa and M.Inaba: "Robot vision system with a correlation chip for real-time tracking, optical flow and depth map generation", in Proc. of the 1992 International Conference on Robotics and Automation (ICRA'92), pp.1621-1626, 1992.

[27] R.A.Brooks: "Intelligence without reason", in Proc. of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), pp.569-595, 1991.

[28] R.C.Arkin: "Motor Schema-Based Mobile Robot Navigation", IEEE Journal of Robotics and Automation, Vol.8, No.4, pp.92-112, 1988.

[29] P.Maes: "The Dynamics of Action Selection", in Proc. of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89), pp.991-997, 1989.

[30] F.R.Noreils and R.Prajoux: "From Planning to Execution Monitoring Control for Indoor Mobile Robots", in Proc. of the 1991 International Conference on Robotics and Automation (ICRA'91), pp.1510-1517, 1991.

[31] R.Hartley and F.Pipitone: "Experiments with the Subsumption Architecture", in Proc. of the 1991 International Conference on Robotics and Automation (ICRA'91), pp.1652-1658, 1991.

[32] M.Watanabe, K.Onoguchi, I.Kweon and Y.Kuno: "Architecture of Behavior-Based Mobile Robot in Dynamic Environments", in Proc. of the 1992 International Conference on Robotics and Automation (ICRA'92), pp.2711-2718, 1992.

[33] M.Watanabe, K.Onoguchi, Y.Kuno and I.Kweon: "Group-based Multi Agent System Configuration for Robot Navigation", Journal

of the Robotics Society of Japan (JRSJ), Vol.13, No.3, pp.375-382, 1995 (in Japanese).

[34] I.Kweon, Y.Kuno, M.Watanabe and K.Onoguchi: "Behavior-Based Mobile Robot using Active Sensor Fusion", in Proc. of the 1992 International Conference on Robotics and Automation (ICRA'92), pp.1675-1682, 1992.

[35] I.Kweon, Y.Kuno, M.Watanabe and K.Onoguchi: "Behavior-Based Intelligent Robot in Dynamic Indoor Environments", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'92, pp.1339-1346, 1992.

[36] A.Kass, A.Witkin and D.Terzopolous: "Snakes: Active Contour Models", International Journal of Computer Vision, No.1(4), pp.321-331, 1988.

[37] I.Kweon, Y.Kuno, M.Watanabe and K.Onoguchi: "Sonar-Based Behaviors for a Behavior-Based Mobile Robot", IEICE Trans.Inf and Syst., Vol.E76-D, No.4, pp.479-485, 1993

[38] N.Ayache: "Artificial Vision for Mobile Robots - Stereo Vision and Multisensory Perception", The MIT Press, ISBN 0-262-01124-7, 1991.

[39] H.Takahashi and F.Tomita: "Planarity Constraint in Stereo Matching", in Proc. of 9th International Conference on Pattern Recognition (ICPR), pp.446-449, 1988.

[40] N.Ayache and O.D.Faugeras: "Building, Registration, and Fusing Noisy Visual Maps", in Proc. of the First International Conference on Computer Vision (ICCV), pp.73-82, 1987.

[41] M.Watanabe, K.Onoguchi, Y.Kuno, H.Hoshino and S.Tsunekawa: "Obstacle detection method for mobile robots with stereo vision", in Proc. of the 5th Scandinavian Conference on Image Analysis (SCIA), pp.325-334, 1987.

[42] N.Ayache and F.Lustman: "Fast and Reliable Passive Trinocular Stereovision", in Proc. of the First International Conference on Computer Vision (ICCV), pp.422-427, 1987.

[43] O.Nakayama, A.Yamaguchi, Y.Shirai and M.Asada: "A Multi-stage Stereo Method Giving Priority to Reliable Matching", in Proc. of the 1992 International Conference on Robotics and Automation (ICRA'92), pp.1753-1758, 1992.

[44] O.Nakayama and Y.Shirai: "Stereo Matching for Occlusion Boundaries Using Normalized Light Intensity", Journal of the Robot Society of Japan (JRSJ), Vol.14, No.7, pp.112-118, 1996 (in Japanese).

[45] M.Herman: "Fast, three-dimensional collision-free motion planning", in Proc. of the 1986 International Conference on Robotics and Automation (ICRA'86), pp.1056-1063, 1986.

[46] A.Elfes: "Using Occupancy Grids for Mobile Robot Perception and Navigation", the IEEE Computer, pp.46-57, 1989.

[47] O.Khatib: "Real-time obstacle avoidance for manipulators and mobile robots", the International Journal of Robotics Research, Vol.5, No.1, pp.90-98, 1986.

[48] E.L.Bras-Mehlman, M.Schmitt, O.D.Faugeras and J.D.Boissonnat: "How the Delaunay Triangulation can be used for Representing Stereo Data", in Proc. of the Second International Conference on Computer Vision (ICCV), pp.54-63, 1988.

[49] E.Huber and D.Kortenkamp: "Using Stereo Vision to Pursue Moving Agents with a Mobile Robot", in Proc. of the 1995 International Conference on Robotics and Automation (ICRA'95), pp.2340-2346, 1995.

[50] H.Nishihara: "Practical real-time imaging stereo matcher", Optical Engineering, Vol.23, No.5, 1984.

[51] M.Watanabe, K.Onoguchi, Y.Kuno and H.Asada: "Obstacle Detection by Disparity Prediction Stereopsis Method", Trans. of

the Institute of Electronics, Information and Communication Engineers (IEICE) D-II, Vol.J73-D-II, No.6, pp.862-870, 1990 (in Japanese).

[52] M.Watanabe, K.Onoguchi, Y.Kuno and H.Asada: "Obstacle Detection by Disparity Prediction Stereopsis Method", Systems and Computers in Japan, 22(10), pp.50-59, 1991.

[53] M.Kidode: "Image processing machines in Japan", IEEE Computer, Vol.16, No.1, pp.68-80, 1983.

[54] K.Onoguchi, N.Takeda and M.Watanabe: "Planar Projection Stereopsis Method for Road Extraction", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'95, pp.249-256, 1995.

[55] V.Graefe: "An Approach to Obstacle Recognition for Autonomous Mobile Robot", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'90, pp.151-158, 1990.

[56] V.Graefe: "Detection of Passing Vehicles by a Robot Car Driver", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'91, pp.391-396, 1991.

[57] G.Adiv: "Determining Three-Dimensional Motion and Structure from Optical Flow Techniques", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 7(4), pp.384-401, 1985.

[58] T.Y.Tian and M.Shah: "Recovering 3D Motion of Multiple Objects Using Adaptive Hough Transformation", in Proc. of the Fifth International Conference on Computer Vision (ICCV)'95, pp.284-289, 1995.

[59] N.Ohta: "Structure from Motion with Confidence Measure and Its Application for Moving Object Detection", Trans. of the Institute of Electronics, Information and Communication Engineers (IEICE), J76-D-II , pp.1566-1571, 1993 (in Japanese).

[60] R.A.Brooks: "Model-based Three-dimensional Interpretations of Two-dimensional Images", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 5(2), pp.140-150, 1983.

[61] Y.Kuno, Y.Okamoto and S.Okada: "Object Recognition Using a Search Strategy Generated from a 3-D Model", in Proc. of the Third International Conference on Computer Vision (ICCV)'90, pp.626-635, 1990.

[62] S.Ullman: "The Interpretation of Visual Motion", Psychological Research, 38, pp.379-383, 1976.

[63] G.Johansson: "Spatio-Temporal Differentiation and Integration in Visual Motion Perception", Cambridge, MA:MIT Press, 1979.

[64] J.K.Aggarwal and N.Nandhakumar: "On the computation of motion from sequences of images - A review", Proc. of the IEEE, Vol.76, No.8, pp.917-935, 1988.

[65] B.K.P.Horn and B.G.Schunck: "Determining Optical Flow", Artificial Intelligence, Vol.17, pp.185-203, 1981.

[66] B.D.Lucas and T.Kanade: "An iterative image registration technique with an application to stereo vision", in Proc. of 5th International Joint Conference on Artificial Intelligence (IJCAI-81), pp.674-679, 1981.

[67] P.Anandan: "A Computational framework and an algorithm for the measurement of visual motion", International Journal of Computer Vision, Vol.2, No.3, pp.283-310, 1989.

[68] D.Heeger: "Optical flow using spatiotemporal filters", International Journal of Computer Vision, Vol.1, pp.279-302, 1988.

[69] D.Fleet and A.Jepson: "Computation of component image velocity from local phase information", International Journal of Computer Vision, Vol.5, No.1, pp.77-104, 1990.

[70] J.L.Barron, D.J.Fleet and S.S.Beauchemin: "Systems and Experiment Performance of Optical Flow Techniques", International Journal of Computer Vision, Vol.12, No.1, pp.43-77, 1994.

[71] S.Yasutomi, H.Mori and N.Kiyohiro: "A Method for Pedestrian Detection Based on Rhythm of Walking", Trans. of the Institute of Electronics, Information and Communication Engineers (IEICE), J78-D-II , No.4, pp.608-617, 1995 (in Japanese).

[72] R.Nelson and Y.Aloimonos: "Obstacle Avoidance Using Flow Field Divergence", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 11(10), pp.1102-1106, 1989.

[73] T.Camus, D.Cooms, M.Herman and T-H.Hong: "Real-Time Single-Workstation Obstacle Avoidance Using Only Wide-Field Flow Divergence", in Proc. of 13th International Conference on Pattern Recognition (ICPR'96), pp.323-330, 1996.

[74] A.Blake and A.Yuille: "Active Vision", ISBN 0-262-02351-2, The MIT Press, 1992.

[75] M.Watanabe, N.Takeda and K.Onoguchi: "Moving Obstacle Recognition by Optical Flow Pattern Analysis for Mobile Robots", Advanced Robotics (The International Journal of the Robotics Society of Japan), (submitting).

[76] J.Weber and J.Malik: "Robust Computation of Optical Flow in a Multi-Scale Differential Framework", in Proc. of the Fourth International Conference on Computer Vision (ICCV)'93, pp.12-20, 1993.

[77] N.Takeda, M.Watanabe and K.Onoguchi: "Moving Obstacle Detection using Residual Error of FOE Estimation", International Workshop on Intelligent Robots and Systems(IROS)'96, pp.1642-1647, 1996.

[78] M.Watanabe, N.Takeda and K.Onoguchi: "A Moving Object Recognition Method by Optical Flow Analysis", in Proc. of 13th International Conference on Pattern Recognition (ICPR'96), pp.528-533, 1996.

LIST OF PUBLICATIONS AND TECHNICAL REPORTS BY THE AUTHOR

1. M.Watanabe, K.Onoguchi, Y.Kuno, H.Hoshino and S.Tsunekawa: "Obstacle detection method for mobile robots with stereo vision", in Proc. of the 5th Scandinavian Conference on Image Analysis (SCIA), pp.325-334, 1987.

2. K.Onoguchi, M.Watanabe, Y.Okamoto and H.Asada: "Visual Navigation System for a Mobile Robot", in Proc. of International Workshop on Intelligent Robots and Systems (IROS)'89, pp.590-597, 1989.

3. M.Watanabe, K.Onoguchi, Y.Kuno and H.Asada: "Obstacle Detection by Disparity Prediction Stereopsis Method", Trans. of the Institute of Electronics, Information and Communication Engineers (IEICE) D-II, Vol.J73-D-II, No.6, pp.862-870, 1990 (in Japanese).

4. K.Onoguchi, M.Watanabe, Y.Okamoto, Y.Kuno and H.Asada: "A Visual Navigation System using a Multi-Information Local Map", in Proc. of the 1990 International Conference on Robotics and Automation (ICRA'90), pp.767-774, 1990.

5. M.Watanabe, K.Onoguchi, Y.Kuno and H.Asada: "Obstacle Detection by Disparity Prediction Stereopsis Method", Systems and Computers in Japan, 22(10), pp.50-59, 1991.

6. M.Watanabe, K.Onoguchi, I.Kweon and Y.Kuno: "Architectureof Behavior-Based Mobile Robot in Dynamic Environments", in Proc. of the 1992 International Conference on Robotics and Automation (ICRA'92), pp.2711-2718, 1992.

7. I.Kweon, Y.Kuno, M.Watanabe and K.Onoguchi: "Behavior-Based Mobile Robot using Active Sensor Fusion", in Proc. of the 1992 International Conference on Robotics and Automation (ICRA'92), pp.1675-1682, 1992.

8. I.Kweon, Y.Kuno, M.Watanabe and K.Onoguchi: "Behavior-Based Intelligent Robot in Dynamic Indoor Environments", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'92, pp.1339-1346, 1992.

9. K.Onoguchi, M.Watanabe, Y.Okamoto and Y.Kuno: "Multi-Information Local Map for Visual Navigation", Journal of the Robotics Society of Japan (JRSJ), Vol.11, No.3, pp.401-409, 1993 (in Japanese).

10. I.Kweon, Y.Kuno, M.Watanabe and K.Onoguchi: "Sonar-Based Behaviors for a Behavior-Based Mobile Robot", IEICE Trans.Inf and Syst., Vol.E76-D, No.4, pp.479-485, 1993

11. M.Watanabe, K.Onoguchi, Y.Kuno and I.Kweon: "Group-based Multi Agent System Configuration for Robot Navigation", Journal of the Robotics Society of Japan (JRSJ), Vol.13, No.3, pp.375-382, 1995 (in Japanese).

12. K.Onoguchi, N.Takeda and M.Watanabe: "Planar Projection Stereopsis Method for Road Extraction", in Proc. of International Workshop on Intelligent Robots and Systems(IROS)'95, pp.249-256, 1995.

13. N.Takeda, M.Watanabe and K.Onoguchi: "Moving Obstacle Detection using Residual Error of FOE Estimation", International Workshop on Intelligent Robots and Systems(IROS)'96, pp.1642-1647, 1996.

14. M.Watanabe, N.Takeda and K.Onoguchi: "A Moving Object Recognition Method by Optical Flow Analysis", in Proc. of 13th International Conference on Pattern Recognition (ICPR'96), pp.528-533, 1996.

15. M.Watanabe, N.Takeda and K.Onoguchi: "Moving Obstacle Recognition by Optical Flow Pattern Analysis for Mobile Robots", Advanced Robotics (The International Journal of the Robotics Society of Japan), (submitting).