MS Thesis Proposal

Behavior-Based Perceptual Navigation Systems For Powered Wheelchair Operations

Hajime Uchiyama
Artificial Intelligence Center
The University of Georgia
Athens, GA 30602-7415
hajime@uga.edu

1. The Problem

The objective of the proposed research is to develop a semi-autonomous architecture of a robotic wheelchair, "a wheelchair agent," that consists of perceptual and navigational capabilities by means of computer science, robotics, and sensory technology. The wheelchair agent aims at improving independent mobility of multi-disabled individuals, and this project focuses on integrating sensory information and human-machine interaction.

The base wheelchair is a standard power chair (Figure 1) that consists of two front pivot wheels, two rear motorized wheels, a battery pack, and a controller (joystick). The perceptual navigation system consists of a computer, a collection of sensors (e.g. ultrasonic, infrared, and CCD camera), and man-machine interfaces (tactile and/or auditory). The behavior-based control architecture used in robotics is expected to be suitable for this architecture because of its modularity and reactivity.

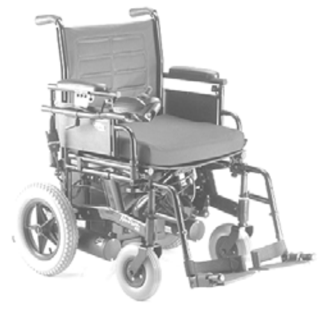


Figure 1: The Power Wheelchair (Invacare Nutron R-32).

2. Background of the Problem

The wheelchair project was started to support Julia Lundy, a visually impaired and temporarily wheelchair-bound student at the University of Georgia. The goal of the project is to establish equipment on her wheelchair that will improve her independent mobility. The interview with Julia in CSCI-6530 (Introduction to Robotics) during fall semester 2002 made us aware of a profound need to conduct research in this challenging field. Further reading revealed that we were joining a small group of pioneers in this relatively new field of research (Gomi and Griffith, 1998; Levine et al, 1999; Yanco, 1998).

Generally any type of assistance to operate a wheelchair varies according to the user's requirement; one might have restricted sensory perceptions, such as limited sight and deafness, and / or impaired motor control. A navigational system must provide as much assistance as the user really needs. Based on the interview, we have assumed the target users have partially limited perception (visually impaired but tactilely and audibly competent with fine motor control of the upper extremities). The inquiry and further discussion have enabled us to elucidate the user needs as the following behavioral units:

- Collision Avoidance (include secure safety while backing up)
- Human Detection
- Dropping-off Avoidance (specifically adjacent to steps)
- Portal Navigation (such as doorways and gates)
- Guiding-information Acquisition (such as signs and room numbers)
- Floor Navigation (to navigate the user to pre-specified destinations by exploiting map and landmark representation)

The first three of those behaviors (Collision Avoidance, Human Detection, and Dropping-off Avoidance) seem to be relatively simple while the others (Portal Navigation, Guiding-information Acquisition, and Floor Navigation) clearly constitute more complicated tasks.

The University of Georgia provides the on-campus curb-to-curb van transportation service to persons with mobility, visual, and other health-related impairments; however, no official care attendant service inside a building is provided. Essentially those behavioral functions for the wheelchair agent are needed mostly indoors. The domains are characterized by dynamic (e.g. obstacles) and partially static (e.g. landmarks) knowledge of the environment.

In this project we aim to examine a hypothesis that tight coupling man-machine interactions aided by the wheelchair agent will increase independent mobility of multi-disabled individuals. Thus, those behaviors will be designed to perform their functionality through user-machine cooperation; the wheelchair agent will interpret the state of the world and pass the information to the user (*Perceptual Behaviors*), and the user will manipulate the wheelchair accordingly in order to achieve the desired goals (*Manipulative Behaviors*).

3. Behaviors

3.1. Perceptual Behaviors

The wheelchair agent is responsible for the perceptual behaviors that will aid the user's limited perception. Let us consider some of the requirements for the perceptual behaviors.

Firstly, all of the perceptual behaviors have to be reflexive and robust. They must deal with a real, complex, and dynamic environment. Any delay or failure of perceptual responses is unacceptable because it may create sever, possibly fatal, consequences for the user.

Secondly, each perceptual behavior unit stimulates human perception through its man-machine interface (an effecter). Each of the signals from the effecter should be intuitive to learn, distinctive from others, and robust against outside noise. We assume that our target users are competent in both tactile and auditory perception; therefore, the output configuration from the perceptual behaviors should convey information (navigational or warning signals) to the user through either tactile or auditory feedback.

Thirdly, the perceptual behaviors should be performed independently in a parallel manner. Since each behavior is defined unitarily, a behavior is responsible for achieving a particular task, based on the sensory inputs that are relevant to its internal process. However, this does not mean we should restrict ourselves to designing only primitive behaviors. Capability of interaction between the behaviors must be provided; some outputs from a behavior may or may not be inputs of another behavior.

Finally, each perceptual behavior should be designed incrementally (modularly) according to its purpose (goal). Some behaviors may consist of other subordinate behaviors, while others may be structured in simplicity, nearly hard-wired to the sensors. An arbitration process will be required in order for the wheelchair agent to accommodate the multiple outputs from the various behaviors. Individual behaviors are prioritized based on pre-determined criteria, such as emergency, significance, or user needs.

3.2. Manipulative Behaviors

The human operator of the wheelchair is responsible for controlling the actual wheelchair movements. Existing hardware on the motorized wheelchair interprets the operator's commands through the joystick and activates the two direct current (DC) motors driving the rear wheels (one motor for each wheel).

The basic scheme for driving the wheelchair is to push the joystick toward the desired direction and to release the joystick to slow down. The joystick housing is located at the front of the right armrest, and the joystick is equipped with 360 degrees of mobility. The joystick is spring-loaded and automatically returns to the upright (neutral) position when released. Pushing the joystick in a given direction causes the chair to move in that direction (Figure 2). The joystick has proportional drive control, meaning that the further it is pushed from the neutral position, the faster the wheelchair moves. To release the joystick causes the wheelchair to slow to a stop. The wheelchair has automatic speed and direction compensation.

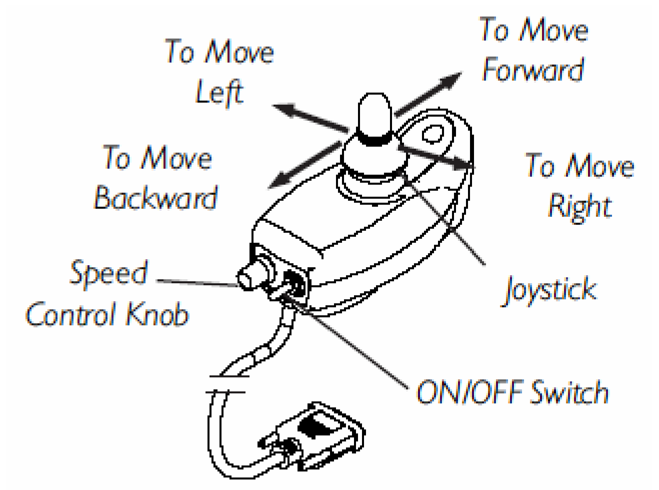


Figure 2: A Schematic Diagram of the Joystick and Wheelchair Operation.

Manipulative behaviors will be achieved such that the operator of the wheelchair recognizes the state of the environment and manipulates the wheelchair with a single command, such as stopping the wheels, or with a sequence of commands. The recognition of the environment is carried out through devices using tactile (and/or auditory) feedback and through her/his own available senses, such as hearing the background noises or acquiring the haptic information by using a cane. The operator will also have to arbitrate if there is a conflict between perceptual behaviors (provided by the wheelchair agent) and her/his own senses.

3.3. Control Architecture Issues in Robotics

In order for the wheelchair navigation to function in real, complex, and unpredictable environments, the perceptual behaviors must perform reasonably and timely. Reference to discussions of robot control architectures may advocate designing the wheelchair navigation system.

Deliberative architectures that rely on a centralized world model operate by collecting all available sensory data, generating a complete model of its static environment, planning an optimal series of actions based on that model, and executing that plan (Nilsson, 1984; Moravec, 1988 and 1989; Laird & Rosenbloom, 1990). However, in such a sense-plan-act paradigm uncertainty in sensing and changes in the environment require frequent re-planning, the cost of which may impede achievement of the goals.

Behavior-Based approaches (Arkin, 1998; Brooks, 1991a and 1991b; Matarić, 1991 and 1992) provide substrata on which perceptual processing is distributed across multiple independent agents (behaviors). Each behavior, operating asynchronously and concurrently, is responsible for a particular task based on the sensory data that is relevant to its process.

The Subsumption Architecture (Brooks, 1986 and 1987) represents behaviors as separate layers and organizes them in a bottom-up manner with a pre-prioritized control framework. Behaviors (layers) work on individual goals, and the lowest (most primitive) layer is defined as an augmented finite state machine (AFSM). The Subsumption Architecture arbitrates among behaviors based on explicitly assigning priorities to each behavior; the output from a behavior with the highest priority is predominant and the rest are ignored. Such priority-based arbitration may be effective for choosing among incompatible outputs; however, due to the absence of the ability to store internal state dynamically, this priority-based arbitration is incapable of either providing any decisions (outputs) that satisfy multiple goals simultaneously or solving complex problems that contain temporal sequences.

The Behavior-Based architecture is founded on the Subsumption Architecture with a capability of containing an internal state and constructing a flexible control framework. Such a capability enables the system to employ various forms of distributed representations implemented within the behavior networks (Matarić, 1997). An abstract behavior representation also allows the Behavior-Based architecture to generate and maintain complex plan-like strategies without redesign and recompilation processes (Nicolescu & Matarić, 2000). Yet having such advantages of flexibility, the methodology of behavior arbitration is one of the challenges of the Behavior-Based architecture.

In order to provide a multiple-goal-oriented decision in the Behavior-Based architecture, it is essential for a behavior arbitration mechanism to obtain *command fusion* functionalities. Command fusion combines the commands (outputs) from individual behaviors so that the decisions may satisfy multiple goals while preserving the reactivity and modularity of the system. Several command fusion approaches have been proposed by combining with a variety of algorithms: the activating spreading schema, centralized voting system from distributed behaviors, fuzzy logic control, immunological network evolved by a genetic algorithm, hybrid automata, and neural network (Maes, 1989; Payton et al., 1992; Yen & Pfluger, 1995; Watanabe et al., 1998; Egerstedt, 2000; Na & Oh, 2003).

4. The Proposed Research

4.1 Behavior Architecture Overview

Behavior Cell Design

Our control architecture methodology of the wheelchair agent is designed based on the Behavior-Based architecture with the extended inputs/outputs feature; we call each unit of the behavioral structure a *behavior cell*. A behavior cell consists of an inputs/outputs (I/O) component, a behavioral function component, and an internal storage component (Figure 3). It structurally seems to resemble an artificial neuron; however, it has a logical gate in addition to widely extended functions such that the innervation link between cells can run by both Boolean and numeric means.

A behavior cell does not have to employ all components; it may or may not consist of the behavioral function and the internal storage components depending upon what features it needs. Behavior cells communicate with other behavior cells, sensors, and effectors through their I/O components.

The I/O component consists of a subset of octagonal I/O ports: Port-EB, excitatory inputs; Port-IB, inhibitory inputs; Port-DI, sensory/behavioral inputs; Port-RS, a reset signal; Port-IS, an innervation signal; Port-AB, an activation output; Port-EO, an effect output; and Port-AO, actuator outputs. The excitatory and inhibitory inputs are linked to the corresponding behaviors' activation output ports. When any activation (inhibition) conditions are met, the behavior is activated (deactivated). Our architecture allows both Port-EB and Port-IB to specify activation (inhibition) conditions by using logical expressions (Boolean algebraic functions), such as:

Activation (or Inhibition) = (Activation_1 OR Activation_2) AND NOT (Activation_3).

Port-DI takes various types of data inputs from sensors or other behaviors (effect outputs). When Port-IS receives an innervation signal from outside or from Port-RS, the behavior checks or sends its inputs and outputs. If Port-RS receives a reset signal from the system, the behavior will clear all dynamic contents of the storage component and activate Port-IS. Port-AB contains an activation value (binary) that is linked to the value of Port-EB. Port-EO contains an effect value that is derived from the behavioral function. If the behavior is connected to its effector(s), Port-AO sends Action Outputs to them. The generic features of each port are summarized in Table 1.

Table 1: The Generic Features of Inputs/Outputs Ports of a Behavior Cell

Port ID	Name	Feature
Port-EB	Excitatory Inputs	Binary value is derived from a predefined logical expression and the connected input values.
Port-IB	Inhibitory Inputs	Binary value is derived from a predefined logical expression and the connected inputs values.
Port-DI	Sensory/Behavioral Inputs	Input data from sensors and/or other behaviors. A various types of the data (including arrays) are allowed.
Port-IS	Innervation Signal	Innervates the behavior cell.
Port-RS	Reset Signal	Clear temporal memory and activate the inner clock.
Port-AB	Activation Output	Binary value linked to Port-EB and Port-IB
Port-EO	Effect Outputs	Outputs (the result of computation)
Port-AO	Action Outputs	Action control to effectors.

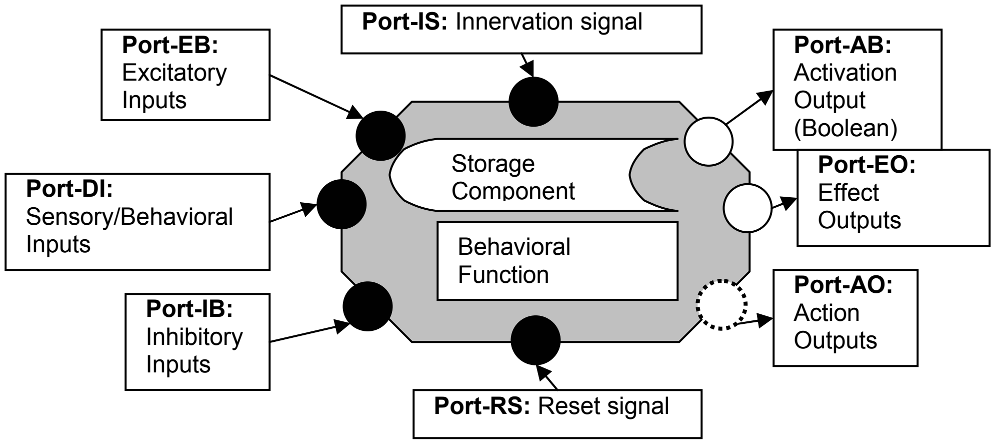

Port-IS: Innervation signal
Port-EB: Excitatory Inputs
Port-AB: Activation Output (Boolean)
Port-DI: Sensory/Behavioral Inputs
Storage Component
Port-EO: Effect Outputs
Behavioral Function
Port-IB: Inhibitory Inputs
Port-AO: Action Outputs
Port-RS: Reset signal

Figure 3: The Basic Structure of a Behavior Cell.

The behavioral function component provides a flexible activation / computation functionality. The function can be various types, such as algebraic sum, sigmoid, Gaussian, and look-up-table, as well as a simple by-pass function (e.g. a direct link between inputs and outputs). More complicated functionalities, such as fuzzy logic inference operators or artificial neural networks, can also be implemented.

The storage component provides a storing capability of the current state onto its dynamic memory, which enables the behavior to achieve goals that contain temporal sequences. It may also contain internal static memory that a behavior utilizes as permanent reference information, such as threshold values, logical expressions (e.g. IF/THEN rule), or look-up tables.

The activation/computation process performed by a behavior cell is as follows:

(0) When Initialization / Reset input (Port-RS) is activated, it refreshes the internal dynamic memory and innerves Innervation Input (Port-IS).

(1) When Innervation Input is innerved, check the value of Effect Inputs (Port-EB). If true, set Activation Output (Port-AB) value to 1 (true) and go to the next step, otherwise return.

(2) Check the value of Inhibitory Inputs (Port-IB) to see whether the behavior is inhibited. If false, go to the next step, otherwise set Activation Output (Port-AB) to 0 (false) and return.

(3) <In case of using Port-EO> Using the information from Sensors / Behavior Inputs (Port-DI), derive the return value from the behavioral function and write this value to Effect Output (Port-EO) and return. Store the necessary data in the internal memory if so designed.

(4) <In case of using Port-AO> Similar to (3), derive the return action commands from the behavioral function and send the commands to the effectors via Action Outputs (Port-AO) and return. Store the necessary data in the internal memory if so designed.

Behavior Network Design

Like other Behavior-Based architectures, our approach also enables behaviors to consist of other behaviors. Such behaviors are implemented based on a subset of corresponding behaviors, thus represented as a *behavior network*. The types of relationships between a behavior network and its corresponding behaviors may vary; they can be hierarchical or interdependent. In a behavior network behaviors communicate with each other through their port-to-port links, and precondition dependence characterizes the links; thus, the activation of a behavior is dependent on its pre-conditional links. The structural flexibility of a behavior cell enables a behavior network to accomplish various tasks, such as command arbitration, learning, and planning. The bottom line of the Behavior-Based philosophy, the distributed architecture, is preserved such that the behaviors are relatively simple, organized into modules, and performed in a parallel fashion.

A behavior network should also work as if it is a behavior cell when observed outside of it. In order to do so, each behavior network is designed to contain a specific type of behavior cell (*I/O cells*), which accomplish the tasks related to inputs/outputs communication and activation sequence inside of the behavior network, in addition to task-oriented / reactive behaviors (*functional behaviors*). Figure 4 depicts a generic behavior network that consists of I/O cells and functional behaviors.

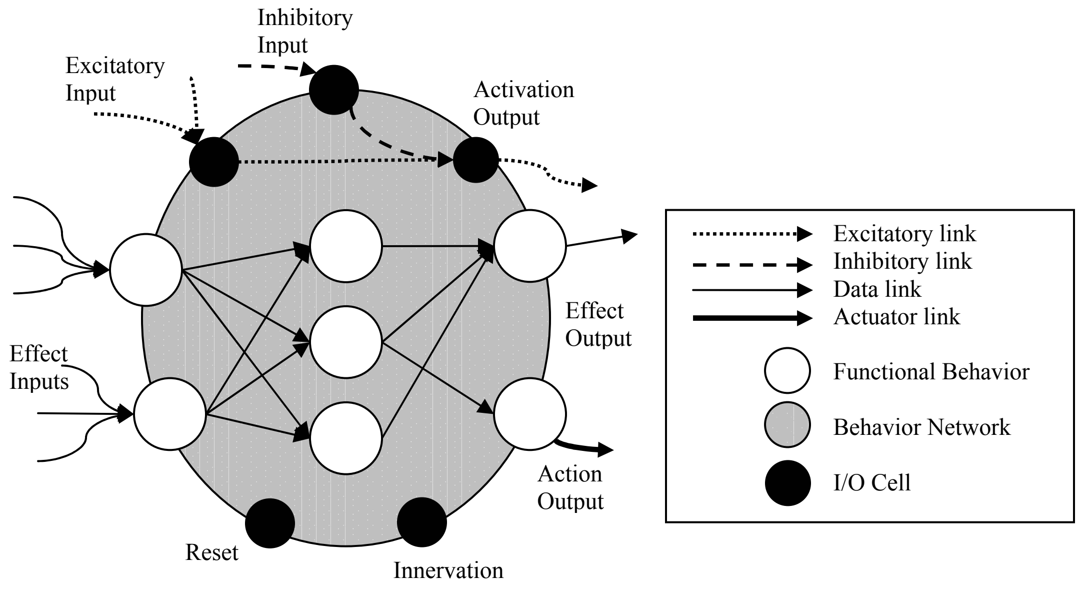


Figure 4: A Schematic Diagram of a Behavior Network.

The I/O cells are categorized in two types: Boolean I/O cells that exchange Boolean signals and activation I/O cells that control sequential activation in the behavior network. Figure 5 illustrates a generic Boolean I/O cell that consists of Port-EB (excitatory inputs), Port-RS (reset signal), Port-IS (innervation signal), Port-AB (activation output), and a Boolean algebraic function. The Boolean algebraic function can employ various symbolic logic expressions, and the result from the function is directly connected to the activation output (Port-AB).

Activation I/O cells are responsible for the sequence of innervation / reset functionalities in the behavior network. An activation I/O cell consists of an excitatory input, an inhibitory input, an innervation input, an activation output, action outputs, and a storage component that contains a predefined activation sequence of the behaviors (Figure 6). The activation I/O cell innerves the reset / innervation ports of the behaviors that belong to the behavior network according to the sequence stored on the storage component.

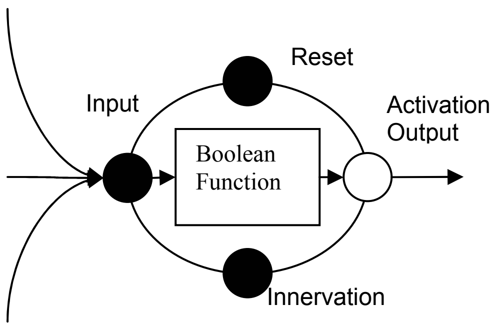


Figure 5: Boolean I/O Cell.

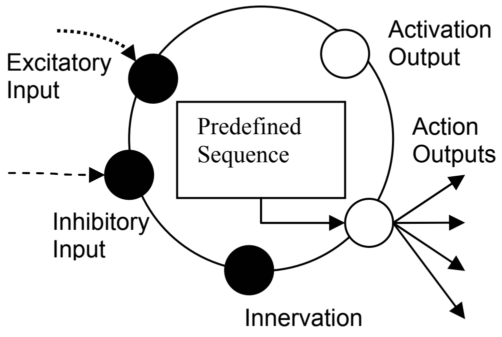


Figure 6: Activation I/O Cell.

The functional behaviors deal with data and/or actuator communication. Connections between functional behaviors consist of excitatory links (between Port-AB and Port-EB), inhibitory links (between Port-AB and Port-IB), (sensory and/or behavioral) data links (between Port-EO and Port-DI), and actuator links (between Port-AO and effectors). A functional behavior in a behavior network can also be another behavior network.

The activation process of a behavior network differs from the one of a behavior cell. An innervation cell is first innerved and it innervates the Excitatory Inputs cell, Inhibitory Inputs cell, and Activation Output cell in this order. If the value of Port-AO (activation output) of an Activation Output cell is true, it will innervate the functional behaviors in a predefined order otherwise the whole process will return; thus the behavior network will be deactivated (Figure 7).

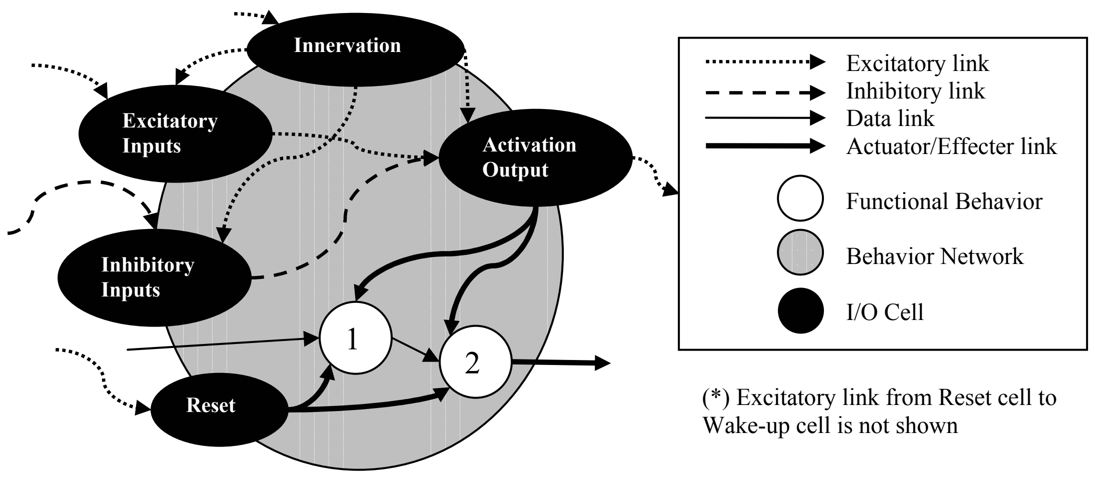


Figure 7: A Schematic Diagram of I/O Cell Links in a Behavior Network.

4.2. Sensor Modules for Perceptual Behaviors

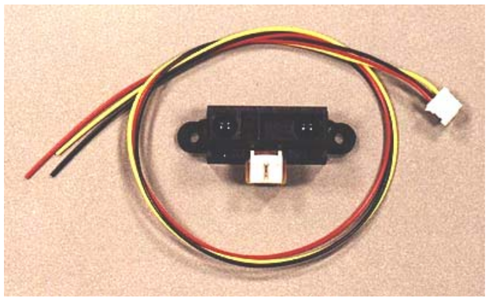
The behaviors on the machine side are perceptual behaviors based on the Behavior Based architecture. The environment information for the perceptual behaviors is acquired by means of a collection of sensors: ultrasonic sensors, infrared (IR) sensors, and CCD cameras.

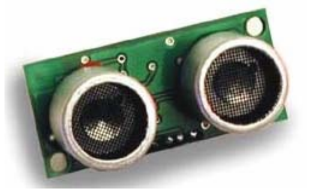
In order to maximize the coverage area with minimum investment for sensory equipment, the sensors and cameras are motorized by a servomechanism to swivel. This feature also enables the sensors to acquire fine spatial resolution of the images from the environment

The ultrasonic and IR sensors actively measure distances of any objects that are within the scope of the sensors; thus, these sensors are called the *Ranging Modules* in this project. The CCD cameras and servomotors (*Motorized Vision Modules*) acquire scenery images, so that the perceptual behaviors can utilize more detailed information, such as the shape of an object in the environment.

Ranging Module:

The ranging module consists of an ultrasonic sensor (Devantech SRF04), an IR sensor (Sharp GP2D12), and a standard servomotor (Hitec HS-300) (Figure 8). The ultrasonic sensor and the IR sensor are coupled in a casing and mounted onto the horn of the servomotor so that they can swivel (Figure 9). The ultrasonic sensor is equipped to face horizontally, and the IR sensor is adjusted to tilt toward the floor in order to detect any objects with low height that might be overlooked by the ultrasonic sensor. The SRF04 ranges from one inch to 10 feet with its beam broadcasted about 25 to 30 degrees wide, and the range of the GP2D12 is from 4 to 32 inches with its beam roughly football shaped with the widest portion in the middle being about 6 inches wide.


(A) Sharp GP2D12 IR Ranger


(B) Devantech SRF04 Ranger

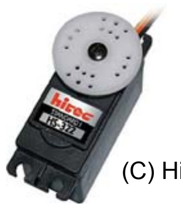

(C) Hitec HS-300 Standard Servo

Figure 8: IR Sensor (A), Ultrasonic Sensor (B), and Servo (C).

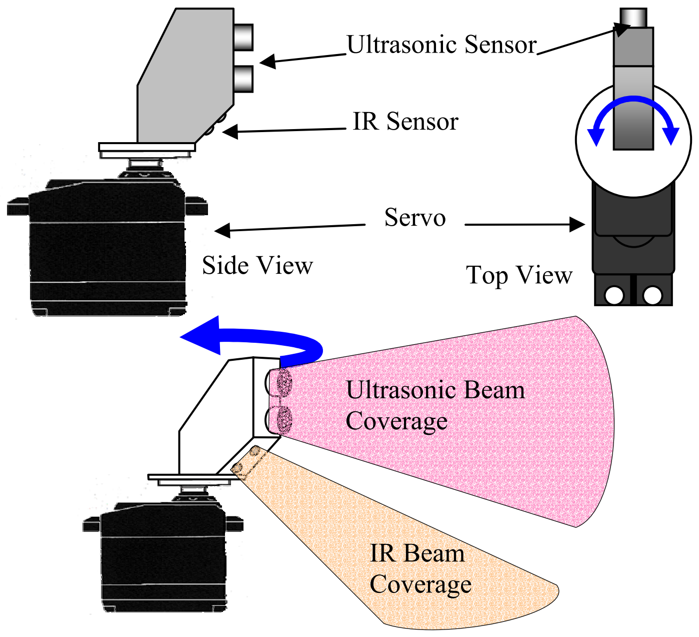


Figure 9: Schematic Diagrams of the Ranging Module and its Beam Coverage.

The HS-300 generally rotates a specific angle according to given pulse signals, pulse-width-modulated (PWM) waves. The angle is proportionally determined by the duration of peak period (high voltage) from 0.5 ms (-90 degrees) to 2.5 ms (+90 degrees) followed by 10 to 20 ms bottom period (low voltage) (Figure 10). The typical speed of rotation is 0.19 sec (4.8 V) or 0.15 sec (6.0 V) per 60 degrees.

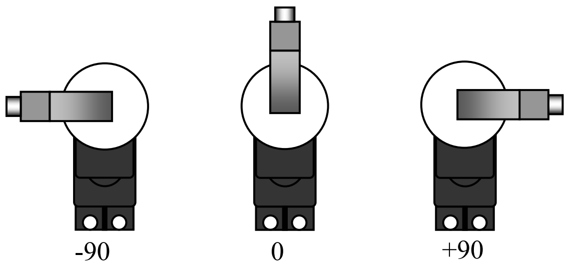


Figure 10: Schematic Diagrams of the Typical Angles of the Ranging Module.

The ranging modules are equipped at the corners of the wheelchair, on both sides of the swing-back arms and the back canes, so that they can provide coverage for large areas around the wheelchair. The schematic diagram of the coverage areas provided by the ranging modules and their installed loci are illustrated in Figure 11.

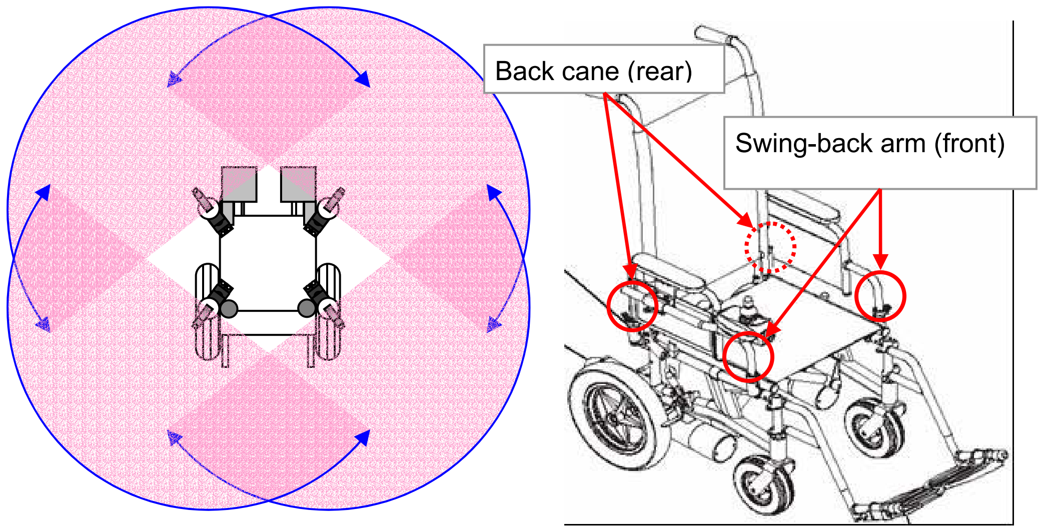


Figure 11: The Coverage Areas and Installation Loci of the Ranging Modules. (The actual size of the ultrasonic coverage area is greater than as it is shown)

Each ranging module can sweep a large coverage area with 180 degrees width; however, the desired coverage area and the format of the distance data may vary depending upon the situation. For example, when the wheelchair is going forward, we may want to acquire not only the closest obstacle data (its distance and angle) for the front side of the wheelchair, but also sensory field mapping data (a series of distance data associated with angles) that may enable the wheelchair agent to make more sophisticated decisions. Obviously the distance data of the backside is not needed as it is going forward, so the rear ranging modules would better check only the side of the wheelchair. On the other hand, when the wheelchair starts turning, obstacles within the particular angles, such as around 45 degrees away from the orientation of the wheelchair on the turning side, and timeliness of detecting them would be more significant in order to prevent collision.

In order to accommodate such needs, we implement two different searching modes: *discrete mapping* mode and *continuous oscillation* mode (Figure 12). The major difference between them is synchronization between sensor activation and the servo movement. In discrete mapping mode, the whole coverage area is broken down into divisions (e.g. the total is 90 degrees wide and one division is 15 degrees wide), and the rotating movement of the servomotor is discrete by division, meaning that it rotates from one division to the next division, stops for ranging, and rotates again. The precision of the distance and angle data matching is guaranteed while the drawback would be sluggishness specifically with many divisions of the coverage area. The data format is a vector containing a series of distance data associated with angle data.

In continuous oscillation mode, the servo movement and sensor activation work independently. The sensors keep ranging distance while the servomotor rotates continuously within the whole coverage area. Continuous oscillation mode is designated for relatively small coverage area when timeliness is required, and the angle of the detected object is not recorded. The data format is scalar whose value is the closest distance data during one oscillation.

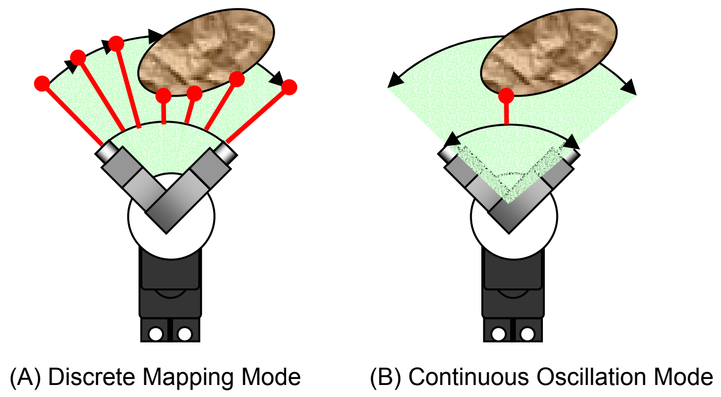


Figure 12: Schematic Diagrams of Discrete Mapping Mode (A) and Continuous Oscillation Mode (B).

The *Range-finder* behavior receives specified angle data and the search mode (discrete mapping or continuous oscillation), controls the servo movement and sensor activation, and sends the sensory readings. The behavior consists of the following subordinate behaviors: *Sensor-reader*, *Synchronizer*, and *Servo-controller* (Figure 13).

The sensing process in discrete mapping mode performed by *Range-finder* is as follows:

(1) *Range-finder* first resets *Synchronizer* and sends angle data and the search mode (discrete mapping mode) to it.

(2) *Synchronizer* first creates a set of discrete angles based on given angle data.

(3) *Synchronizer* sends the target angle to *Servo-controller,* waits until the movement feedback from *Servo-controller* becomes *false* (the servomotor has stopped), and then innervates *Sensor-reader*. During this step,

   a. *Servo-controller* interprets the angle to a PWM wave, sends it to the servomotor, and sets the movement feedback to *true*.
   b. Meanwhile, *Servo-controller* monitors the voltage of the servomotor (assuming that the motor voltage increases while rotating).
   c. When the voltage drop is caught, it sets the movement feedback to *false*.

This process illustrates only one set of ranging module (*Sensor-reader* and *Servo-controller*); however, we have a total of four sets of ranging modules. When multiple ultrasonic sensors that are closely located emit sound waves simultaneously, a sensor may receive erroneous echoes emitted from neighboring sensors (*crosstalk* phenomenon). In order to avoid the phenomenon, we also employ a time-sharing schema; each *Synchronizer* controls a pair of ranging modules that are diagonally mounted on the wheelchair (Figure 14), and the two *Synchronizers* are activated alternately by inserting some duration (about 2 ms) between the activations. The length of the duration is adequate for sound waves to travel the maximum range (10 feet) of the SRF04 and return.

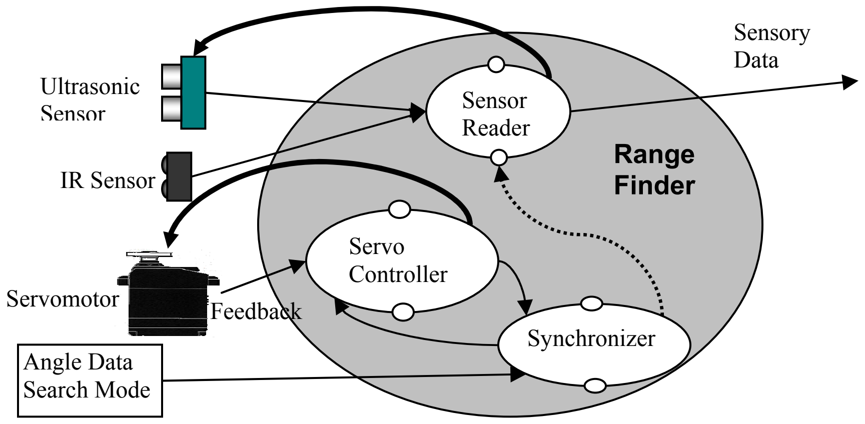


Figure 13: A Schematic Diagram of Range-finder Behavior.

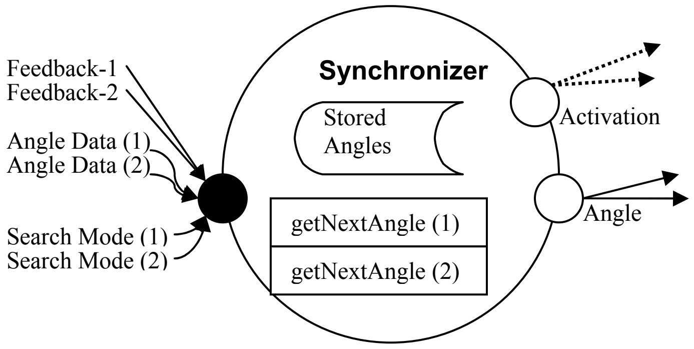


Figure 14: A Schematic Diagram of Synchronizer Behavior.

Motorized Vision Module:

The motorized vision module consists of a CCD camera (Logitec: QuickCam Pro 4000) attached to the horn of a servomotor (Hitec: HS-300) that rotates a specified angle by a given PWM wave. In order to acquire angle and depth information from stereoscopic visionary images, two vision modules are equipped on the bars that are attached to the back canes of the wheelchair and adjusted to be slightly higher than the human operator's eyes (Figure 15). The servo enables the camera to swivel so that the horizontal field of view of the camera is expanded from 42 degrees (the specification of the QuickCam Pro 4000) to over 180 degrees.

The mobility of the motorized vision module was inspired by an ethological hint, such as eye movements of a chameleon. When both cameras face to the same direction and acquire similar images (stereoscopic mode), the depth and distance information of an object of interest can be calculated based on the disparity of the two images. When the cameras work individually, they would function like surveillance cameras, such as following an object and swiveling periodically.

The *Image-getter* behavior receives a specified angle data (-90 to 90 degrees) and sends image data (pixels) acquired by the CCD camera (Figure 16).

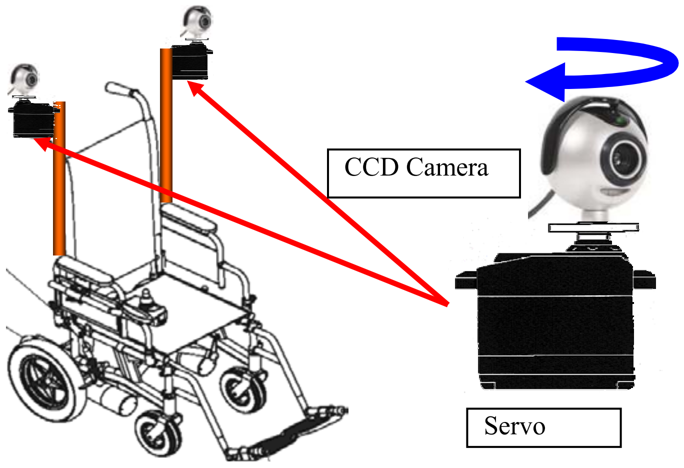


Figure 15: Motorized Vision Module.

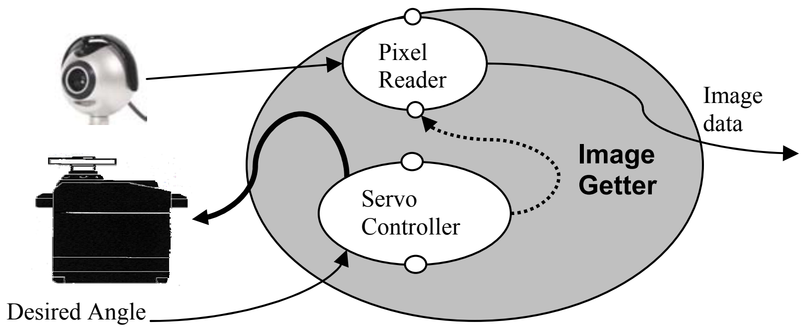


Figure 16: A Schematic Diagram of Image-getter Behavior.

Machine Vision Operation:

The machine vision operations we employ contain image processing, feature extraction, stereoscopic vision, and feature tracking. Short descriptions of those operations are summarized as follows (Jain et al., 1995):

- **Image Processing**: The purpose of image processing is to eliminate unnecessary details and enhance the area of interest in the image for computational efficiency. In this project, we specifically focus on the shape (points and lines) of the object; thus, image processing includes the following sub-operations:
    - Image downsizing
    - Image gray scaling
    - Image filtering
    - Image edge detection
    - Image thresholding
    - Image thinning
- **Feature Extraction**: In feature extraction the area of interest is retrieved from a given image, such as a doorframe and a corridor intersection. This is a cognitive operation, so we will provide heuristic schemata that should help to find such feature points and lines within a reasonable amount of time. In this project we specifically aim to identify a line with a specified inclination with the edge of the doorway or with the corner of the corridor.
- **Stereoscopic Vision**: The purpose of stereoscopic vision is to obtain the accurate spatial relationship between the target object and the wheelchair. If any

point in the scene is visible in both cameras, it will be projected to a pair of points (a *conjugate* pair) in the two cameras. Based on the displacement between the positions of the points (*disparity*) and the known camera factors (such as the distance between the two cameras and the focal length of the camera), the distance and angle of the point would be calculated. The stereoscopic vision approach generally includes a correspondence problem (how to identify a conjugate pair). In this project, we attempt to implement a behavior employing the stereo vision technique that only correlates the features that are retrieved by the feature extraction operation.

- **Feature Tracking**: Coupled with a servomechanism, a behavior employing the feature tracking operation will work like an autonomous agent. This operation enables the behavior to select the feature of interest from the candidate features provided from the feature extraction operation and track the horizontal movements of the feature frame by frame. Once a feature is "locked on", it sends the angle data that rotates the servomotor for the countermovement of the feature. Feature tracking aims to lock on the target in the center of the sight. If the feature moved too fast to follow timely, the amount of the countermovement could be predicted based on differential equations.

4.3. Interface Modules for Perceptual Behaviors

The outputs from the perceptual behaviors should consist of various kinds of vibrotactile signals conveying information via human tactile perception. Previous work evaluating a variety of tactile feedback systems has shown substantial potential (Geldard, 1975; Gemperle et al., 2001; Tan, 1997 and 2000).

Vibrotactile Glove:

As an interface to the blind user, we propose a glove equipped with tactile effectors (Vibratactile Glove). The Vibrotactile Glove consists of an array of vibrating motors shaping a semi-circle (an arc) on the back side of the hand (Figure 17). A motor is located in the center of the arc, and other motors are located along the arc. The vibrotactile array generates individual and directional pulse patterns which are tactilely noticeable but not audibly noticeable. A vibrating motor generates an individual pulse, the feature of which is determined by controlling intensity of the signal, duration of pulses, and the interpulse interval. A directional pattern consists of the pulse generated by the central motor (marked “C” in Figure 17) followed by another pulse from one of the circled motors (from “A1” to “A5” in Figure 17). The individual pulse will be used for notifying the user of an obstacle’s locus, and the directional patterns are designed to indicate the preferable direction to the user.

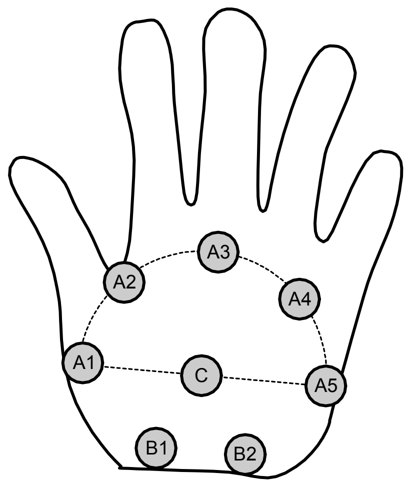


Figure 17: The arrayed motors of the Vibrotactile Glove

Audio Devices (audio alarm):

Since our target users are audibly competent, utilizing audio devices, such as speakers, headphones, and microphones, will enhance the capability of conveying detailed information from the machine to the user. However, one of the disadvantages of audio devices is that the effects of the devices may vary depending upon the surrounding audio noise level. Some situations may not allow the user to use any audio devices either. Thus, audio devices should be considered as optional or secondary interface modules.

Audio alarms via speakers (or an open-air headset) generated by the user's PC will enable the machine to catch the user's attention. By varying the pitches and the patterns of the tones, the machine can send simple information, such as a middle tone with short duration period twice for 'good,' a high tone with middle duration for 'caution,' and a low tone with long duration for 'dangerous.' Implementation of alarm patterns is simple and does not require any additional resources.

4.4. Implementation of Perceptual Behaviors

The essential goal of the perceptual behaviors is to provide navigational and/or warning guidance to the human operator through tactile and/or audible interfaces. The perceptual behaviors are designed for indoor usage with partial domain knowledge, such as destination and portal information. The specifications of the goals / tasks performed by the behaviors are as follows:

(1) While roaming without any specific instructions from the operator, be responsible mainly for object notification; however, also search a landmark or some orientation that seems to be appropriate in that situation and periodically navigate that direction. For example, when the wheelchair is located in a corridor, navigate the user to follow

the corridor orientation. If no features of the corridor are detected (meaning that the wheelchair may be located in a big hall), look for an exit and notify the direction to the user.

(2) When a destination (goal) has been specified and a known landmark is detected, create a plan from the current location to the destination by using given topological knowledge (planning) and give directional guidance to the user. This directional guidance must take care of the obstacle notification task.

(3) If the current landmark is a portal (such as a doorway, a corridor intersection, or a gate) and the plan is to pass through the portal, navigate the user to do so with more accurate guidance than other navigations.

In order for the wheelchair agent to accomplish these goals, we implement the following behaviors: two reactive behaviors, *Obstacle-notification* and *Roaming-explorer*; and a task-oriented behavior, *Portal-navigation*, which also includes a command arbitrating behavior, *Navigator,* based on a fuzzy logic control.

In this proposal, we do not endeavor extensively to accomplish specification (2) by implementing a complex planning behavior; however, we do utilize the framework by implementing a planner-like behavior (*Floor-navigation*), which executes hardwired plans.

4.4.1 Obstacle-notification

Obstacle-notification performs one particular function: notifying the user of any nearby obstacles in order for the user to avoid contact with them. *Obstacle-notification* acquires the obstacle information and activates/deactivates the Vibrotactile glove accordingly. It is implemented as a behavior network and consists of three nearly reactive behaviors, *Obstacle-detection*, *Translator*, and *Vibrotactile-motor-controller* (Figure 18). *Obstacle-notification* is a default behavior, meaning that it is activated initially when the system is turned on or when other task-oriented behaviors accomplish their tasks. It accepts inhibitory signals from *Portal-navigation* and *Floor-navigation*.

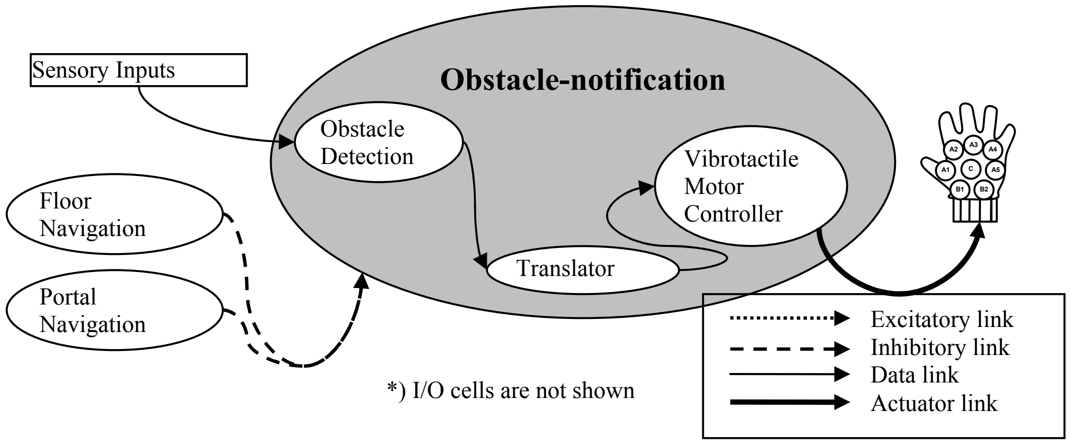


Figure 18: A Schematic Diagram of Obstacle-notification.

Obstacle-detection

Obstacle-detection consists of *Range-finder-director*, *Sensor-direction-calculator*, and *Data-organizer* (Figure 19). *Sensor-direction-calculator* receives the moving direction data, determines coverage area for each ranging module, and sends that information to the *Range-finder-director*. *Range-finder-director* takes sensory inputs form desired angles and sends the inputs to *Data-organizer*. *Data-organizer* disjunctively selects the closest distance data at each angle when multiple objects are detected in the scan angle. It also fuses the inputs from the IR and ultrasonic sensors. The output is an array of vectors, each of which contains direction and distance data about the detected object.

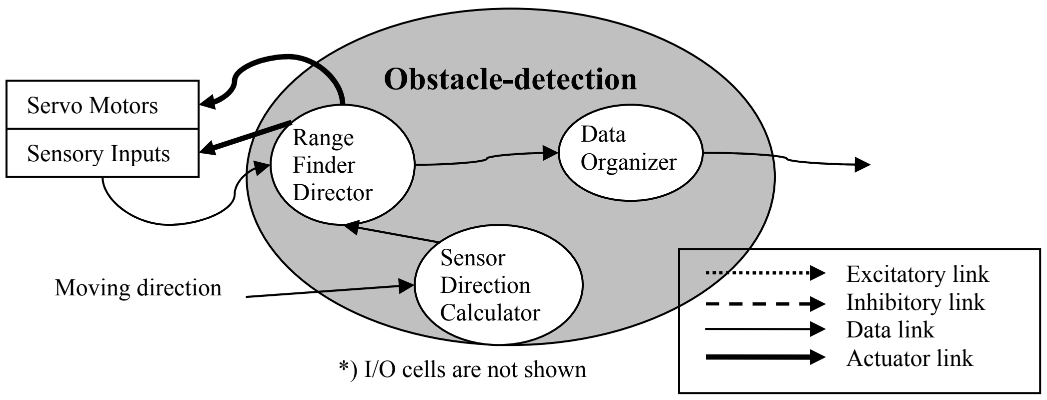


Figure 19: A Schematic Diagram of Obstacle-detection.

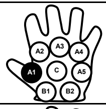
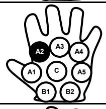
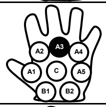
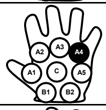
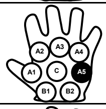
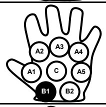
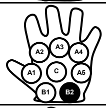
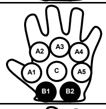
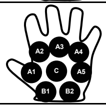
Translator

Translator receives the input from *Obstacle-detection* and sends a command to *Vibrotactile-motor-controller*. It determines the state of surrounding obstacles based upon the internal IF/THEN rules and look-up tables. Then it translates the state into a command based on a protocol of the motor controller, such as a unique integer. If any proximate objects are detected within a predefined threshold, it will also generate a 'stop' sign (all motors activation).

Vibrotactile-motor-controller

Vibratactile-motor-controller receives the output signal from *Translator* and generates the actual vibration pattern for the vibrating motors. The duration of the vibration and the magnitude of the vibration are determined according to the distance from the detected object; that is, the closer the object is, the more duration and magnitude the vibration will have. If the output signal contains the 'stop' sign, the warning signal combines the 'stop' signal with the corresponding motor activation alternately with the maximum magnitude. Typical warning signals from the *Vibratactile-motor-controller* behavior are shown in Table 2.

Table 2: Warning signals for the Vibrotactile glove (Obstacle-notification).

Meaning	Activated motor(s) Vibrating pattern	Motor locus ● = activated
Object detected at left	A1 Intervallic	
Object detected at front-left	A2 Intervallic	
Object detected at front	A3 Intervallic	
Object detected at front-right	A4 Intervallic	
Object detected at right	A5 Intervallic	
Object detected at rear-left	B1 Intervallic	
Object detected at rear-right	B2 Intervallic	
Object detected at rear	B1 and B2 Intervallic	
Stop	All motors Intervallic	

4.4.2 Roaming-explorer

The purpose of *Roaming-explorer* is to provide appropriate direction to the operator in the absence of any specific instructions. It consists of the following behaviors: *Corridor-detection*, *Landmark-detection*, *Obstacle-detection*, *Navigator*, *Translator*, and *Vibrotactile-motor-controller* (Figure 20). *Roaming-explorer* accepts inhibitory signals from *Portal-navigation* and *Floor-navigation*. The process of this behavior is as follows:

(1) When the user is roaming without a specific purpose, periodically have the CCD cameras take the image of the scene.

(2) If *Landmark-detection* finds a doorframe within the image, register the direction and go to step (5), otherwise go to step (3).

(3) If no doorframes are found, but *Corridor-detection* determines that the wheelchair is located in a corridor, register the direction for following the corridor and go to step (5), otherwise go to step (4).

(4) Swivel the cameras to obtain a different angle of the scene, and go to step (2).

(5) *Obstacle-detection* receives the registered direction, searches any nearby obstacles on its way, and sends that information to the next behavior.

(6) *Navigator* receives the suggested direction and the obstacle information, and arbitrates the command by utilizing the fuzzy logic technique. Then it sends a crispy command that controls the Vibrotactile glove activation.

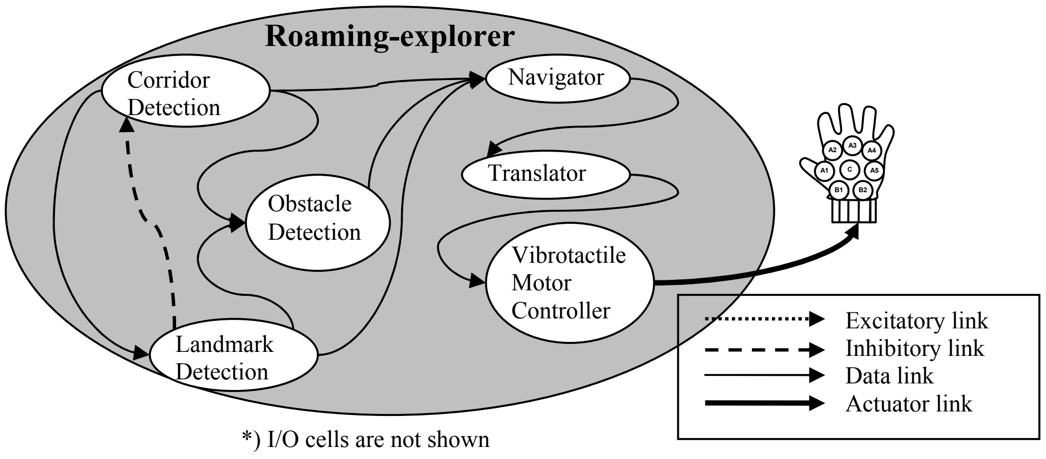


Figure 20: A Schematic Diagram of Roaming-explorer Behavior.

Landmark-detection

Generally *landmark detection* (recognition) aims to identify an object that represents certain attributes (criteria) with a landmark. The attributes can be retrieved by means of a collection of sensors and/or machine vision. If there are several candidate landmarks in the sensory/image field, the landmark detection will be fundamentally equivalent to an analysis of the pattern recognition for classification of objects, which is beyond the aim of this project.

To simplify the pattern recognition part, we define restrictively the domain of a landmark as only doorframes supplemented by corridor lines. They consist of lines with certain angles retrieved by visionary images (Figure 21). In case multiple doorframes are detected, the selection strategy will be either (1) the doorframe with the closest distance (binocular vision) or (2) the doorframe with the closest angle (monocular vision).

The *Landmark-detection* behavior receives preprocessed images (pixels) and sends the positioning data of the detected objects (a pair of vertical lines of the doorframe), such as angle and distance (if possible) from the centroid of the wheelchair. The data format is a vector (angle and distance), which can also be represented as polar coordinates. Primarily the closer distanced line of the object is registered as a reference object, whose locus in the pixel image will also be sent to the next behavior. If distance measurement is unavailable (monocular vision mode), the line on the same side as the tracking camera (if the right camera is tracking the doorframe, the right vertical line) will be registered.

Landmark-detection consists of several vision-based behaviors: *Feature-extraction*, *Corridor-detection*, *Doorframe-detection*, and *Depth-calculation*. All of these behaviors are based on the machine vision techniques described earlier.

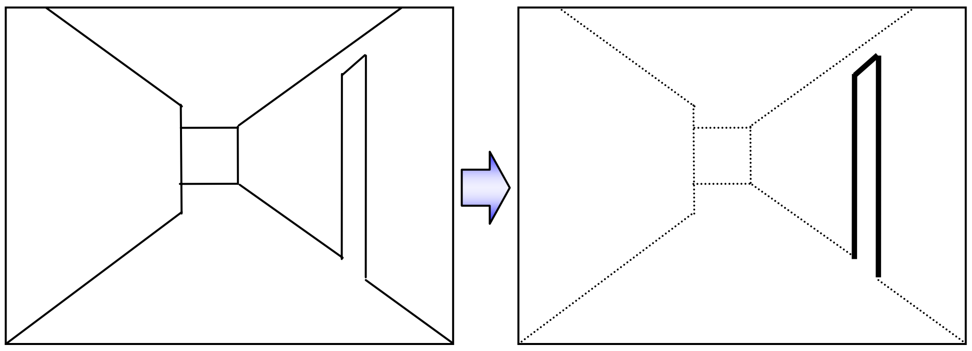


Figure 21: A Schematic Diagram of Doorframe Extraction.

Corridor-detection

Corridor-detection is also a vision based behavior that receives preprocessed images (pixels). If it detects a corridor, it will send the orientation of the corridor from the center of the wheelchair. The determination process of a corridor is performed by searching a pair of lines that are restricted by predefined inclinations in the scene (Figure 22). A Hough transformation technique may be utilized in order to detect the candidate lines and their inclinations.

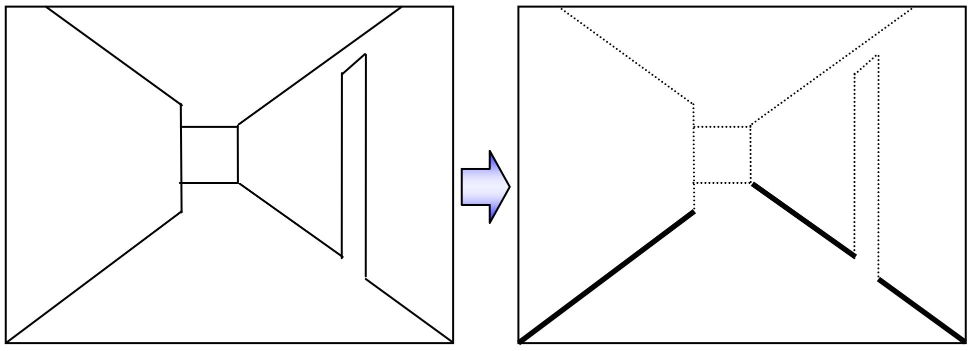


Figure 22: A Schematic Diagram of Corridor Extraction.

Navigator

Navigator receives a desired direction and obstacle information, and generates a compromised output (direction). We take a similar approach for command fusion as done in the previous work (Rosenblatt, 1997a and 1997b; Yen & Pfluger, 1995). *Navigator* contains several behavior cells based on fuzzy logic techniques: *Fuzzy-interface* for obstacle information and the suggested direction, *Command-fusion*, and *Defuzzification* (Figure 23).

The *Fuzzy-interface* behavior contains a set of fuzzy rules and a fuzzy inference module. It first transforms specific direction data into a linguistic representation, which gives more flexibility in avoiding obstacles while moving toward a landmark. *Command-fusion* combines obstacle information and suggested direction, and generates a fuzzy control command. It uses the MIN operator that constructs a conjunction of the directional data input. *Defuzzification* provides the process of converting a fuzzy command into a crisp command (e.g. "turn front-left"). The most appropriate defuzzificaiton methodology will be empirically chosen from the Mean of Maximum method, the Center of Area method, or the Centroid of Largest Area (Yen & Pfluger, 1995).

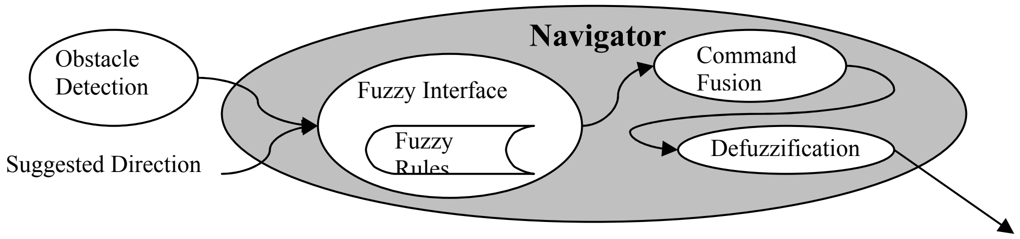
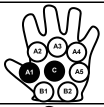
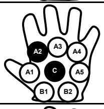
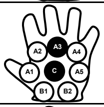
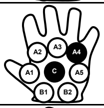
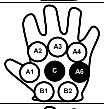
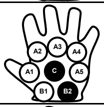
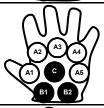
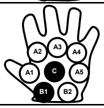


Figure 23: A Schematic Diagram of Navigator Behavior.

Vibrotactile-motor-controller

In addition to the signals described in Table 2, *Vibrotactile-motor-controller* also generates a directional signal that suggests an appropriate direction to move. These directional signals are shown in Table 3.

Table 3: Directional Signals for the Vibrotactile Glove.

Meaning	Activated motor(s) Vibrating pattern	Motor locus ● = activated
Go left (left swiveling)	C followed by A1 Intervallic	
Go front-left	C followed by A2 Intervallic	
Go forward	C followed by A3 Intervallic	
Go front-right	C followed by A4 Intervallic	
Go right (right swiveling)	C followed by A5 Intervallic	
Back-off rear-right	C followed by B2 Intervallic	
Back-off	C followed by B1 & B2 Intervallic	
Back-off rear-left	C followed by B1 Intervallic	

4.4.3 Portal-navigation

The goal of *Portal-navigation* is to navigate the wheelchair to pass through a portal, such as a doorway, a corridor intersection, and a gate. Even with competent visual perception, going through a doorway itself may not be an easy task for a wheelchair operator as we drive a car more slowly at a gate than on a road because more accurate control is required at the gate.

The process of passing through a doorway can be divided into several subtasks. The wheelchair operator needs to (1) realize that the door is the portal, (2) determine whether the door is available (opened), (3) if so, cope with closeness to objects, such as walls and poles, (4) adjust an approach angle toward the door, and finally (5) go forward. *Portal-navigation* is fully responsible for task (1) and task (2), and partly for task (3) and task (4).

These tasks are, in short, to determine a maneuvering trajectory for the wheelchair dynamically, and a human driver is capable of performing these two tasks seamlessly. The trajectory of the wheelchair is a result of decision making that relies on high level cognitive processes. The cognitive processes retrieve a variety of information: external static information (such as the location and the opening gap of the entrance), external dynamic information (such as existence of any obstacles nearby), internal static restrictions (such as the size of the wheelchair and its minimum turning radius), and internal dynamic information (such as the approach angle and the distance toward the entrance).

Let us consider some effects of the dynamic factors on the trajectories. Assume that the wheelchair has been following the right-hand side of the wall, and now the portal entrance is detected. The arrow in Figure 24 illustrates a maneuvering trajectory tracing the kinetic centroid of the wheelchair with enough distance from the wall. If the wheelchair has been following the wall too closely, the driver would be better off to gain some distance from the edge of the opening before giving a turn (Figure 25) in order to avoid collision with the edge.

Some of the previous works attempted to create a local map in order to generate a maneuvering trajectory for an autonomous vehicle (Patel et al., 2002; Surmann et al., 2001). The maneuvering trajectory may shape a complicated curvature, and it is unintuitive to represent such a curvature as a sequence of tactile signals. Timeliness and simplicity are essential features of perceptual notifications to the user. Furthermore, we should anticipate that there would be a gap between the ideal trajectory and the actual movement performed by the user. Therefore, *Portal-navigation* is also responsible for compensating at any point of the guidance when it detects the differences between the current and ideal position of the approaching path.

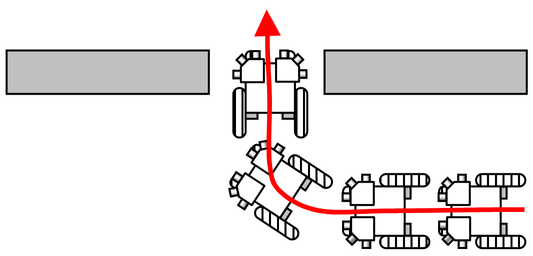


Figure 24: A Typical Maneuvering Trajectory of the Wheelchair.

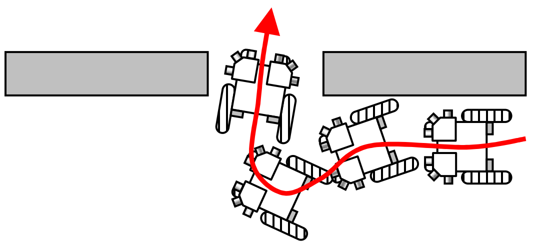


Figure 25: A Maneuvering Trajectory Affected by Closeness between the Wheelchair and the Wall.

In order for our wheelchair agent to accomplish these highly cognitive and dynamic tasks as promptly, accurately, and robustly as possible, we propose a guidance approach that utilizes the *Pivotal Point (Zone)* navigation strategy combined with several machine vision techniques, such as image processing, stereoscopic vision, and feature tracking.

<Portal Site and Pivotal Zone>

When a portal entrance is detected by *Landmark-detection*, closeness of the wheelchair to the portal is ensured, in other words, the wheelchair is in a *portal site*. However, the simplicity and easiness of entering the doorway is not guaranteed yet.

A Pivotal Zone is an auxiliary area from which the wheelchair can move toward the portal entrance straightforwardly. It is located in the portal site and predetermined based on characteristics of individual portals. Pivotal Zones provide an adequate space in which the wheelchair can swivel/turn without hitting any previously known objects (such as door edges and the other side of wall) and a direct orientation toward the portal entrance for the wheelchair. A Pivotal Zone can be calculated if the distance from both edges of the entrance and the orientation of the wheelchair are known. Another important feature is that the portal entrance is always detectable from the Pivotal Zone by means of sensory inputs or visionary images.

This is how it should work by utilizing the Pivotal Zone. As illustrated in Figure 26, when the wheelchair detects the portal site at position (A), it first aims to the Pivotal Zone, position (B). When it reaches the Pivotal Zone, it starts swiveling (clockwise in this example) until it finds the correct direction toward the center of the entrance and moves toward the entrance. The pivotal trajectory is composed of a series of nearly straight lines (the solid arrows) and a sharp turning point instead of a continuous curved line (the dotted arrow) representing a maneuvering trajectory. Even though the traveling length of the pivotal trajectory may be longer than the maneuvering trajectory, the navigational signals for the pivotal trajectory remain simple and intuitive, such as "go front-left," "swivel clockwise," and "go forward."

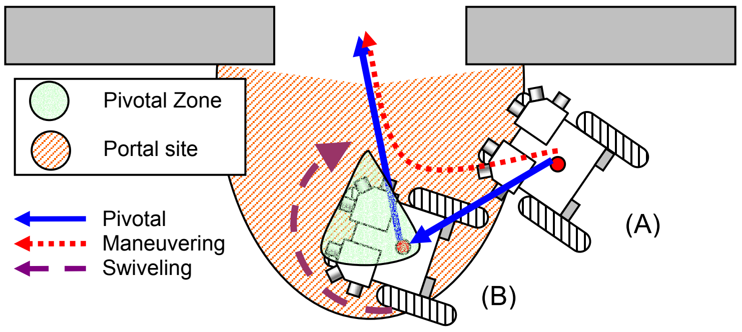


Figure 26: A Pivotal Zone in a Portal Site.

By utilizing this Pivotal Zone strategy and the machine vision techniques, we shall be able to design *Portal-navigation* that consists of the following subordinate behaviors: *Landmark-detection*, *Pivotal-localization*, *LDPP-calculator*, *Adjustment-planner*, *Obstacle-detection*, *Translator*, and *Vibrotactile-motor-controller* (Figure 27). The overview of the process of *Portal-navigation* is as follows:

(1) When *Landmark-detection* finds a portal site, initially a pivotal point whose distance is closest to both door edges (LDPP: Least Distanced Pivotal Point) will be calculated by *LDPP-calculator*. The least distance is equivalent to the minimum turning radius of the wheelchair. (See Figure 31)

(2) *Landmark-detection* also sends the reference coordinates to *Pivotal-localization*.

(3) *Pivotal-localization* tracks the reference object and calculates the current coordinates of the wheelchair.

(4) *Adjustment-planner* receives the LDPP coordinates and the current coordinates of the wheelchair. Coupled with *Obstacle-detection*, *Adjustment-Planner* searches a sequence of the wheelchair operations that navigates the wheelchair to the pivotal zone. If it determines that it is unavailable to turn at the portal entrance, halt the whole navigation.

(5) The directional command from *Adjustment-planner* is sent to the Vibrotactile glove through *Translator* and *Vibrotactile-motor-controller*.

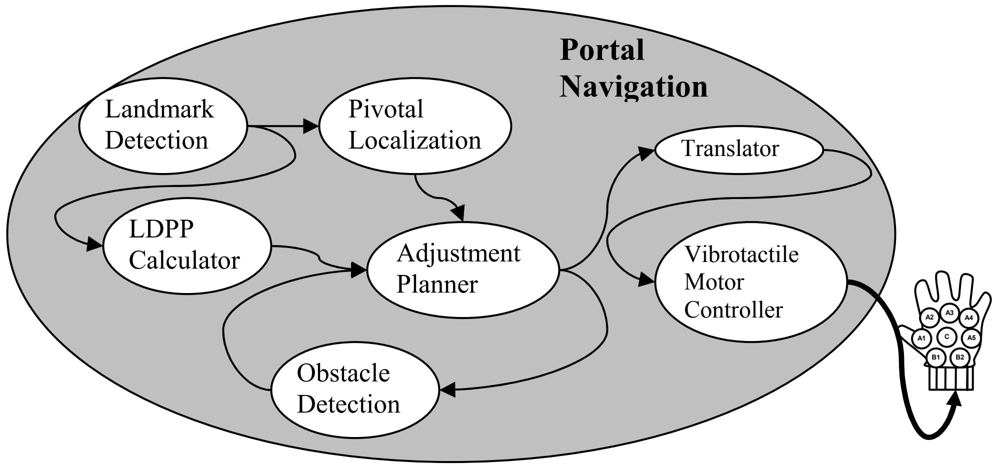


Figure 27: A Schematic Diagram of Portal-navigation Behavior.

Pivotal-localization

The major purpose of the *Pivotal-localization* behavior is to determine whether the wheelchair is located in a Pivotal Zone. It consists of several subordinate behaviors: *Tracking-coordinator*, *Vertical-line-tracer*, *Range-finder*, and *Position-data-calculator* (Figure 28). The position data (distance and direction) of the wheelchair relative to the reference point (localization) is determined either by using two CCD cameras (*binocular* mode) or by using a CCD camera and an ultrasonic sensor (*monocular-sonar* mode). The reference point is to be set as an origin of the local polar coordinate system. The process of localization performed by *Pivotal-localization* is as follows:

(1) Receiving the reference coordinates (the position of the door edge), *Tracking-coordinator* determines which camera(s) should be used for tracking and sends angle data to *Vertical-line-tracer*.

(2) *Vertical-line-tracer* traces the target line and sends its angle data (in binocular mode, it sends two angle values from both cameras) (Figure 33).

(3) With binocular mode, *Position-data-calculator* calculates the angle and distance data of the reference object based on a stereoscopic vision technique. If one of the cameras looses sight of the target, the sonar sensor will start facing nearly the same angle to measure the distance from the target (monocular-sonar mode, Figure 34). In either case, *Position-data-calculator* gives precise coordinates of the centroid of the wheelchair.

Tracking a target by using a combination of different source sensors (e.g. camera and sonar) enables the system to accomplish precise localization of the observer

(Maeyama, Ohya, & Yuta, 1998; also see Figure 35). In our approach the vision-based behavior, *Pivotal-localization*, should be able to track any object in nearly 360 degrees sight by switching the source of the sensors.

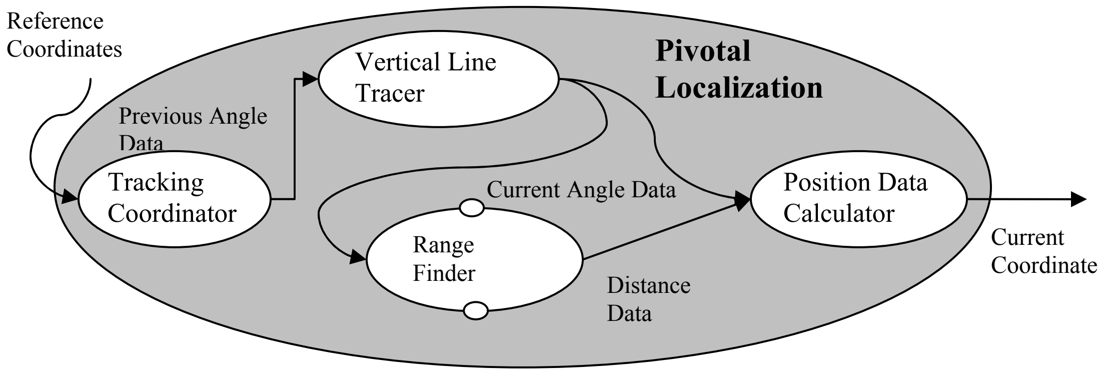


Figure 28: A Schematic Diagram of Pivotal-localization Behavior.

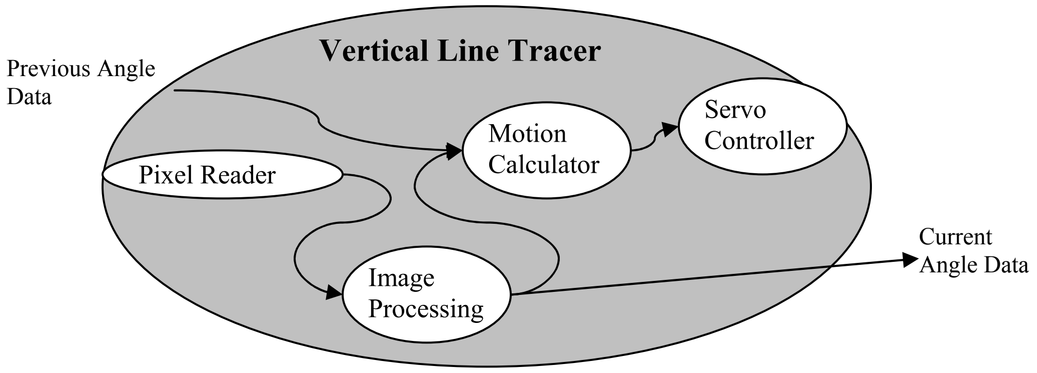


Figure 29: A Schematic Diagram of Vertical-line-tracer.

Vertical-line-tracer

Vertical-line-tracer consists of *Pixel-reader* and *Image-processing*, *Motion-calculator*, and *Servo-controller* for cameras (Figure 29). *Image-processing* acquires the pixel data from *Pixel-reader* and retrieves the current angle data of the target object. *Motion-calculator* receives the previous and current angle data and sends the rotation angle for countermovement to *Servo-controller*.

Adjustment-planner

Adjustment-planner consists of *Planner, Navigator,* and *Validation.* It receives the LDPP coordinates, the current coordinates, and the obstacle data provided by *Obstacle-detection* (Figure 30). The *Planner* behavior creates a sequence of wheelchair movements in order to eliminate the difference between the LDPP and current coordinates (an adjustment path). It provides the optimal adjustment path that contains the smallest amount of adjustment moves. However, the adjustment path may conflict with obstacle avoidance. In order to accommodate such a case, when the *Validation* behavior detects invalid results from *Navigator*, it tells *Planner* to provide an alternative adjustment path.

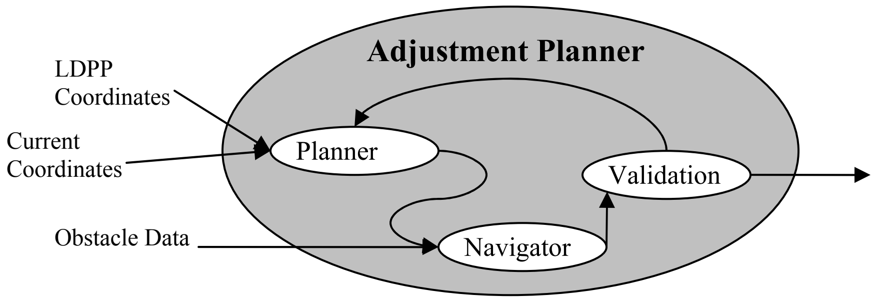


Figure 30: A Schematic Diagram of Adjustment-planner Behavior.

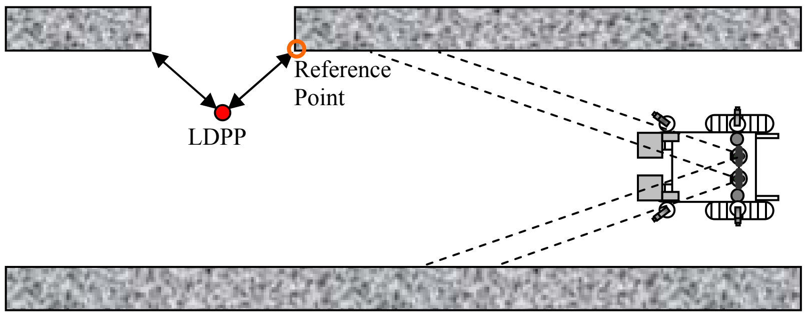


Figure 31: Landmark-detection detects the door as a portal, and the LDPP is calculated. Both door edges are in the sights of both cameras.

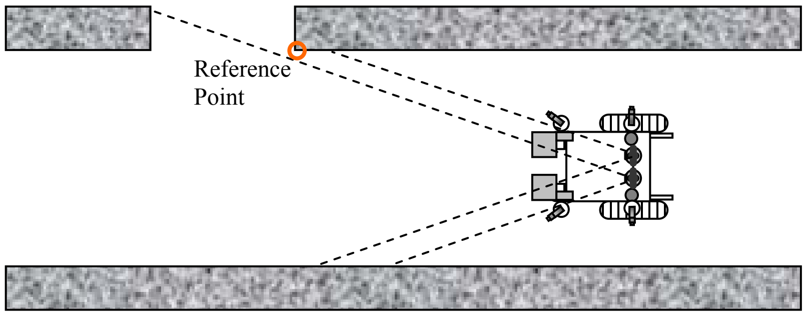


Figure 32: The left camera lost the reference point (the right edge of the door).

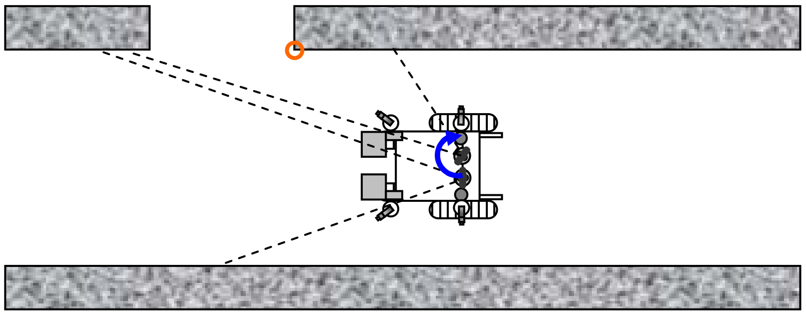


Figure 33: The right camera and sonar start swiveling to track the reference point.

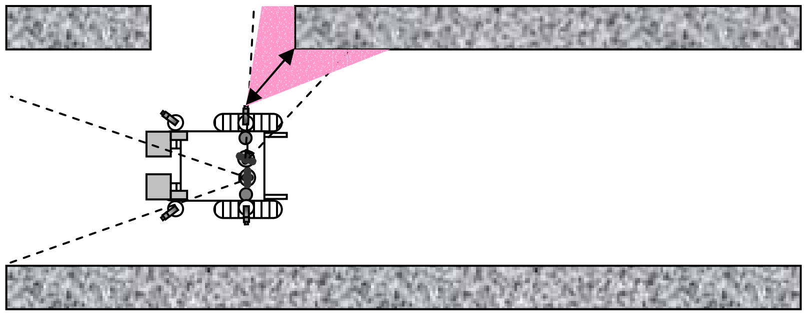


Figure 34: The right sonar also swivels to the same angle of the camera. The sonar turns into continuous oscillation mode, and measures the distance from the closest object (presumably the door edge).

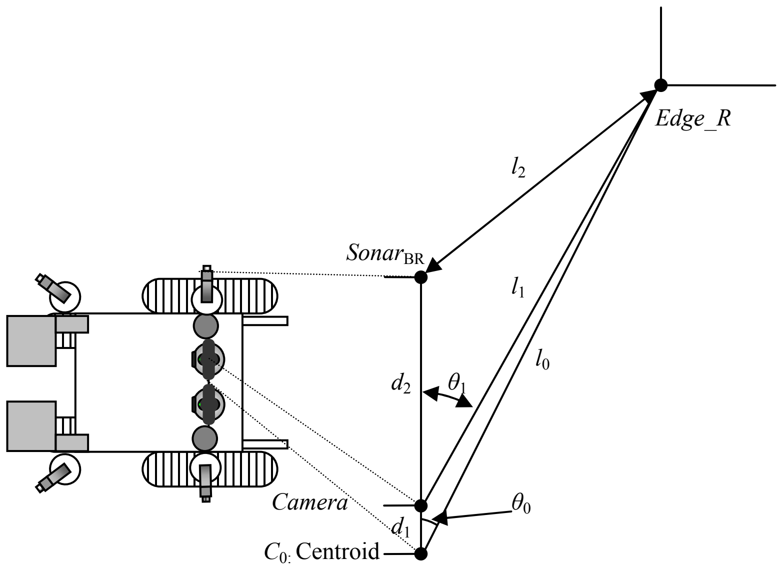


Figure 35 A Schematic Diagram of Dimensions of Local Positioning of the Wheelchair.

4.5. Experiment Procedures Plan

Scheduling

- Pilot Experiment: Installing the CCD camera and servo system onto ER-1 and test how it works (from current to early Jan. 2004).
- Hardware Implementation (from early Jan. 2004 to mid. Feb. 2004)
- Software Implementation (from early Jan. 2004 to end Feb. 2004)
- Modification and writing thesis (from begin Mar. 2004 to end Apr. 2004)

*) Comprehensive Examination will be scheduled on the end of January or on the beginning of February.

Software Implementation

For the actual implementation, Java will be employed as the main programming language and Micro Assembler will also be used for the controller of the vibrotactile displays. All digital inputs will be connected to the laptop PC, and the PC connects to the vibrotactile contoroller via USB / Serial cable.

Experimental Procedure

The feasibility of this design will be examined by several volunteer test drivers (with eyes-shut) to try to maneuver around the AI center at the University of Georgia (the first floor, at Boyd) and pass through the automatic door for wheelchair driver on the second floor. The more advanced examination will be to navigate the wheelchair from the entrance hall to the restaurant in the Georgia Center at the UGA.

4.6. Facilities Needed

For this project, the following equipment is needed:

- A powered wheelchair: Invacare Nutron R32
- Ultrasonic sensor (for ranging sensors, an array module of ultrasonic sensors will be attached to both sides of the swing-back arms of the wheelchair)
    - Devantech SRF04 Ultrasonic Range Finder: four pieces
- Infrared sensor
    - Sharp Infrared sensor GP2D12: four pieces
- Servo motor
    - Hitec HS-300: six pieces
- CCD camera
    - Logitec QuickCam Pro 4000: two pieces
- A laptop computer
- The vibrotactile glove

References

Arkin, R. C., (1998). *Behavior-based robotics.* The MIT Press: Cambridge, Mass.

Brooks, R. A., (1986). "A Robust Layered Control System for a Mobile Robot." *IEEE Journal of Robotics and Automation*, RA-2, April, pp. 14-23.

Brooks, R. A., (1987). "Intelligence without Representation." *Artificial Intelligence*. 47, pp. 139-159.

Brooks, R. A., (1991a). "How to Build Complete Creatures Rather than Isolated Cognitive Simulators." In K. VanLehn (ed.), *Architectures for Intelligence*, pp. 225-239, Lawrence Erlbaum Associates, Hillsdale, NJ.

Brooks, R. A., (1991b). "Integrated Systems Based on Behaviors." *SIGART Bulletin* 2, 2(4), pp. 46-50.

Egerstedt, M., (2000). "Behavior Based Robotics Using Hybrid Automata," *Lecture Notes in Computer Science: Hybrid Systems III: Computation and Control*, pp. 103-116, Pittsburgh, PA, Springer-Verlag, March.

Geldard, F.A., (1975). *Sensory Saltation: Metastability in the Perceptual World*, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Gemperle, F., Ota, N., and Siewiorek, D., (2001) "Design of a Wearable Tactile Display," *Proc. of fifth International Symposium on Wearable Computers (ISWC '01) IEEE Computer Society*.

Gomi, T. and Griffith, A., (1998). "Developing Intelligent Wheelchairs for the Handicapped," in *Assistive Technology and Artificial Intelligence*, V. Mittal, H. Yanco, J. Aronis, and R. Simpson, Eds. New York: Springer, 1998, pp. 151-178.

Jain, R., Kasturi, R., and Schunck, B. G., (1995). *Machine Vision.* McGraw-Hill: New York.

Laird, J. E. and Rosenbloom, P. S., (1990). "Integrating, Execution, Planning, and Learning in Soar for External Environments," *Proc. of AAAI-'90*, pp. 1022-1029.

Levine, S. P., Bell, D. A., Jaros, L. A., Simpson, R. C., Koredn, Y., and Borenstein, J., (1999). "The NavChair Assistive Wheelchair Navigation System," *IEEE Trans. Rehab. Eng*., vol. 7, pp. 443-451, Dec.

Maes, P., (1989). "Dynamics of Action Selection." *Proc. of International Joint Conference on Artificial Intelligence*, Detroit, MI, pp. 991-997.

Maeyama, S., Ohya, A., Yuta, S., (1994). "Positioning by Using Tree Detection Sensor and Dead Reckoning for Outdoor Navigation of a Mobile Robot." *International*

Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'94), Las Vegas, Dec.

Matarić, M. J., (1991). "Behavioral Synergy without Explicit Integration." *SIGART Bulletin* 2, 2(4), pp. 130-133.

Matarić, M. J., (1992). "Behavior-Based Control: Main Properties and Implications." *Proc. of IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, Nice, France, May, pp. 46-54.

Matarić, M. J., (1997). "Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior." *Journal of Experimental and Theoritical Artificial Intelligence* 9(2-3), pp. 323-336.

Moravec, H. P., (1988). "Sensor Fusion in Certainty Grids for Mobile Robots." *AI Magazine, 9(2)*, pp. 61-74.

Moravec, H. P., (1989). "Autonomous Mobile Robot Research Using the HErmies-III Robot." *IORS International Conference on Intelligent Robot and System*, Tsukuba, Japan, Sept.

Na, Y.-K. and Oh, S.-Y., (2003). "Hybrid Control for Autonomous Mobile Robot Navigation Using Neural Network Based Behavior Modules and Environment Classification." *Autonomous Robots*, 15(2), pp. 193-206.

Nilsson, N., (1984). *Shakey the Robot*, SRI Tech. Note 323, Menlo Park, California.

Nicolescu, M. N. and Matarić, M. J., (2000). "Extending Behavior-Based Systems Capabilities Using an Abstract Behavior Representation." In *Working Notes of the AAAI Fall Symposium on Parallel Cognition*, North Falmouth, MA, Nov. 3-5, pp. 27-34.

Patel, S., Jung, S-H, Ostrowski, J. P., Rao, R., and Taylor, C. J., (2002). "Sensor Based Door Navigation for a Nonholonomic Vehicle," *Proc. of the 2002 IEEE International Conference on Robotics and Automation,* Washington. DC. May.

Payton, D., Keirsey, D., Kimble, D., Krozel, J., and Rosenblatt, J. K., (1992). "Do Whatever Works: A robust approach to fault-tolerant autonomous control." *Journal of Applied Intelligence*, (3), pp. 226-249.

Rosenblatt, J. K., (1997a). "Behavior-Based Planning for Intelligent Autonomous Vehicles." In *AV Symposium on Intelligent Autonomous Vehicles, Madrid, Spain.*

Rosenblatt, J. K., (1997b). "Utility Fusion: Map-Based Planning in a Behavior-Based System." *Proc., of FSR `97 International Conference on Field and Service Robotics.*

Surmann, H., Lingemann, K., Nüchter, A., and Hertzberg, J., (2001). "A 3D Laser Range Finder for Autonomous Mobile Robots," *Proc. of the 32nd ISR (International Symposium on Robotics)*, pp. 153-158, 19-21 April.

Tan, H. Z. and Pentland, A., (1997). "Tactual Displays for Wearable Computing," *Proc., of the International Symposium on Wearable Computers*.

Tan, H. Z., Lim, A., and Traylor, R., (2000). "A Psychophysical Study of Sensory Saltation with an Open Response Paradigm," *DSC- Vol. 69-2, Proc. of the ASME Dynamic Systems and Control Division*.

Yanco, H. A., (1998). "Integrating Robotic Research: A survey of robotic wheelchair development." *AAAI Spring Symposium on Integrating Robotic Research*, Stanford Univ., CA, 1998.

Yen, J. and Pfluger, N., (1995). "A Fuzzy Logic Based Extension to Payton and Rosenblatt's Command Fusion Method for Mobile Robot Navigation," *IEEE Transactions on Systems, Man, and Cybernetics, 25(6)*, pp. 971-978, June.

Watanaba, Y., Ishiguro, A., Shirai, Y., and Uchikawa, Y., (1995). "Emergent Construction of Behavior Arbitration Mechanism Based on the Immune System," *Proc. of ICEC'98,* pp. 481-486, 1998