Robotany: Autonomous Vehicles that Care for Houseplants

by

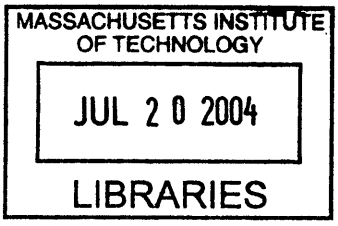
Sara Elizabeth Cinnamon

B.S. Mechanical Engineering
University of California at Berkeley, 2001

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2004



Signature of Author: _____
Department of Mechanical Engineering
January 16, 2004

Certified by: _____
Jean-Jacques E. Slotine
Professor of Mechanical Engineering and Information Sciences
Professor of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by: _____
Ain S. Sonin
Chairman, Department Committee on Graduate Students
Department of Mechanical Engineering

Robotany: Autonomous Vehicles that Care for Houseplants

by

Sara Elizabeth Cinnamon

Submitted to the Department of Mechanical Engineering
on January 16th, 2004 in Partial Fulfillment of the
Requirements for the degree of Master of Science

Abstract

Robotany is a system of autonomous robots that act on behalf of houseplants that rest on top of their chassis. Their duty is to do what plants would if they had the gift of mobility – namely to seek out sunlight or water when there are insufficient amounts of either at their current location. Despite the specialized application, the underlying framework of the robots is rather general and can be used in a variety of situations.

The robots are designed to be easily modifiable for a given application. They are constructed using rapid-prototyping techniques that allow them to be built quickly and inexpensively. A novel design is utilized for the vehicle's suspension. This design is far simpler, cheaper, and more easily customized than traditional systems that perform the same task.

The software controlling Robotany utilizes a behavior-based approach, one that takes its cue from nature's solutions to problems facing any mobile being. It follows Braitenberg's model for seeking out light in an implicit manner. A new approach to obstacle avoidance is used, based on reactance to *in situ* sensor readings and a simplified internal map of the local environment. Robotany also incorporates a simple homeostatic system to regulate the quality of its behaviors and to determine when one behavior should take precedence over another.

Experimental results presented in this thesis show that the robots are successful in finding sources of light while avoiding obstacles in their path.

Thesis Supervisor: Jean-Jacques E. Slotine
Title: Professor of Mechanical Engineering and Information Sciences
Professor of Brain and Cognititve Sciences

Acknowledgements

I would like to thank Bakhtiar Mikhak first and foremost, because if it weren't for him and his continued support, this project and this degree never would have happened. Thank you, Bakhtiar, for being a better advisor in five minutes than others could be in a lifetime.

Thank you Barun, for encouraging me and loving me and for telling it like it is, even if I don't want to hear it. Ooh, and for the banana pancakes too.

To my parents who gave me the life I have, who put me on the right path and always wanted what was best for me. All I've ever wanted is to make you proud. Thank you for everything.

Thank you to Professor Slotine, for adopting me as an advisee and encouraging me to take a vacation somehwere sunny.

To Scooby, for answering silly questions and being so damn smart, Tim, for having incredible response times on emails regarding other silly questions, and Dan for working his Rabbit voodoo on short notice. You guys rock.

Thank you to Wayne, who made my whining seem like nothing. And for the amazing backrubs, of course.

And thank you to all of my friends and family for being who you are. You make life worth living.

"I love deadlines. I like the whooshing sound they make as they fly by."
-Douglas Adams

Contents

List of Figures

Chapter 1

Introduction and Motivation

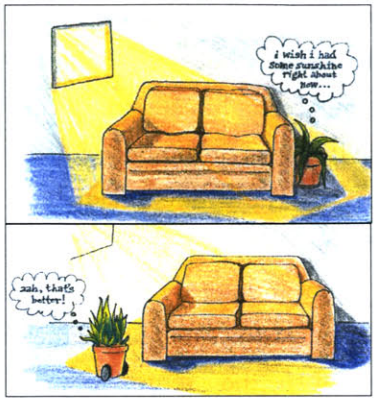


Figure 1.1: A dramatization of the plight of most houseplants.

Imagine a scenario where houseplants can take care of themselves. No longer reliant upon their human caretakers for survival, they roam about their home, interacting with each other and competing for resources. They cluster around the windows during the day, and return to their individual decorating locations in the evening. When the humans come home after work, they are greeted by happy and healthy green plants.

Robotany is a system of artificially intelligent robots that care for houseplants by acting as the plants would if they were able. A plant is situated atop an individual robotic vehicle that is able to move autonomously about the environment. Each robot is then programmed with the needs of the particular houseplant on top of it, such as if the plant requires full sun versus partial shade, and if it needs to be kept evenly moist or prefers to dry out before being inundated with water. Various sensors on board the vehicle are used to monitor the plant's state, allowing the system to know when the plant needs water or light. If some aspect of the plant's state is not at the desired level, the system will use principles of homeostasis to take action to restore balance. For example, if the plant has not received enough sunlight (based on light-sensor readings), the robot will actively search for areas of greater brightness. This search will be more or less aggresive depending on how far the system has strayed from homeostatic balance.

The motivation for creating Robotany can be found throughout homes, offices and public indoor spaces. In all of these settings, plants can often be found clustered near windowsills or placed only in rooms with abundant natural light. When moved away from such locations, houseplants slowly wither from lack of light. While a designer may wish to integrate plants with the décor for aesthetic reasons, they are not able to do so out of consideration for the plants' health. Robotany was conceived to provide a solution to the

houseplants' plight. This solution applies to large plants as well, as Robotany's physical form can be scaled up to accommodate heavy loads.

Robotany is intended to function as an unobtrusive part of the home environment. The robots are designed so that a houseplant can be placed securely on top. An owner can place multiple plants, with Robotany vehicles underneath, anywhere in the environment without concern for relative locations of windows. The robots maintain a timer that can be set to indicate when the owner leaves for work and when they expect to return. In the meantime, the robots are free to navigate the environment on their quest for sunlight for their symbiont plants. Prior to the owner's return, they return to their origins. (The returning mechanism has not been implemented in this thesis, although methods to do so are discussed in section 4.2.2.) This would ensure a seamless operation that minimizes interference with the owner's daily life, while making the care of the plants automatic.

There are many aspects to Robotany's behavior, and its final makeup was influenced by a wide variety of topics within the fields of artificial intelligence, controls, biology, and behavioral psychology. Preferred methods were those that were most closely modeled after nature. Such methods are usually highly efficient approaches to solving the task at hand and require a minimal amount of computational resources. Included in the preferred methods is Braitenberg's synthetic vehicle that seeks out light by way of direct connections between sensors and motors. Breazeal's robot Kismet also influenced the development of Robotany through its use of homeostasis to regulate its behaviors. Similarly, Warren's work in developing a dynamic model of steering in humans served as a guide for what natural navigation in obstacle avoidance should look like. Each of these subjects, as well as a number of others, is discussed in detail in Chapter 2, Theoretical Background.

Robotany's hardware was designed and built specifically for this application. The hardware was designed to be modular, however, to

be useful for any number of projects that required a mobile robot. The details of these modular components can be easily modified for the particular application at hand, and can be scaled to accomodate larger loads. Off-the-shelf robotic solutions were considered, but were either too expensive or had insufficient computation and interface abilities. The first revision of Robotany was implemented using a LEGO™ Mindstorms kit, running only a light-following routine and a simple obstacle avoidance routine using bump sensors. The Mindstorms kit was sufficient for a proof of concept, but more computational power and a greater number of sensor inputs were needed to realize the full potential of the project. Additionally, the structure of the robot needed to be constructed from something more robust to time than LEGO™ bricks. More detail on the structure of the robot is presented in Chapter 3, Implementation.

To satisfy the computational requirements of this project, each Robotany vehicle takes advantage of the Tower System, a development of the Grassroots Invention Group at MIT's Media Lab. This system is designed to enable rapid electronics prototyping with a set of easily extensible tools. The technology behind the Tower can be applied to a full range of application and by a range of users, from novice to expert. A main advantage of using the Tower System over another electrical and programming architecture is smooth and straightforward implementation. For example, the details behind transmitting raw voltages from a sensor to a computer program for use in calculations is simple and well-defined. Another advantage lies in the programming language used in the development environment. RabbitLogo, built upon the Logo language and Rabbit C, uses a straightforward interface to provide high-end functions to speed prototyping and reduce the likelihood of assembly-level errors in the code. For Robotany, this attribute proved particularly useful as the main focus of the project was to develop a functioning robot, not to implement serial protocols or memory-managing routines. The Tower System is discussed in greater depth in Chapter 3. For complete reference, please see Christopher Lyon's thesis on the topic. *[49]*

The fourth chapter of this thesis, Testing and Analysis, presents data on the current state of Robotany's functionality, and provides comparisons between Robotany's behavior and the behavior expected as the result of some of the approaches presented in the background section. It also discusses future work necessary to make the project completely autonomous.

CHAPTER 2

THEORETICAL BACKGROUND

The following sections provide background information on the technologies that influenced the creation of Robotany. This is not meant to be an exhaustive coverage of the information available, but what was most prevalent and widely accessible.

2.1 Behavior-Based Artificial Intelligence

Behavior-based artificial intelligence (AI) approaches computer programming in a way that tries to emulate the complex behaviors exhibited by creatures in nature.*[12]* Such programs do this by reacting to the environment directly through raw sensor readings. They also create starightforward connections among elements of the program, such as between sensor inputs and motor outputs.*[6]* This is in contrast to the classical view of AI where all actions are explicitly programmed, as much information about the environment must be known as possible, and extensive rules are constructed to govern behavior.*[12]* The computer program in the classical case plans out actions based on sensor input and *a priori* knowledge, such as maps of the environment. The program then directs the motors to act according to the decision made. This process can often take some time, depending on the complexity of the task at hand. For example, Honda's bipedal robot ASIMO *[3]* is able to perform complex tasks such as climbing stairs and turning in place. However, these actions are highly structured programs that depend strongly on the details of the environment, which the robot "knows" beforehand. If the width of one stair step were changed, the robot likely would be unable to cope and would topple. A behavior-based program, on the other hand, takes in sensor readings, and relates them to motor output values to produce an immediate change in behavior. The latter approach often results in more "realistic" behavior, and can better respond to dynamic, unstructured environments. Instead of using predetermined maps, behavior-based programs

build their knowledge base in real time by exploring and testing the environment.

Previous research in the field of AI has often focused on computer simulations of desired behaviors and "evolved" solutions in purely simulated environments before implementing the programs on actual robots in the physical world.*[70]* This led to problems, however, because the programs relied on "perfect" sensors and actuators. They were also unable to accurately model the dynamics of locomotion involved or the effects of noise and interference in an unstructured environment. One solution that researchers took was to construct special environments, which allowed the robots to sense only what their programmers wanted them to. This approach eliminated many of the problems previously encountered when moving from simulation to a physical robot, but solutions that were optimized to these conditions still would not work in the "real world."

For Robotany, steps were taken to avoid the problems that crop up when starting with simulations or specially constructed environments. From the beginning, the robots were tested with real (sometimes imperfect) sensors and motors, and in real environments, such as around the Media Lab and my apartment. In the debugging process some factors would be constrained, such as using a lamp to simulate sunlight while turning off obstacle avoidance, but everything was tested in a coherent system to see what, if any, interference would occur between different competing behaviors. Below, the work of three influences is described in further detail, including how they specifically inspired Robotany.

2.1.1 Machina Speculatrix

W. Grey Walter was a research scientist in the mid-20th century specializing in neurophysiology. His background in electrical engineering aided advancement of the use of the electroencephalogram to study the "black box" that the

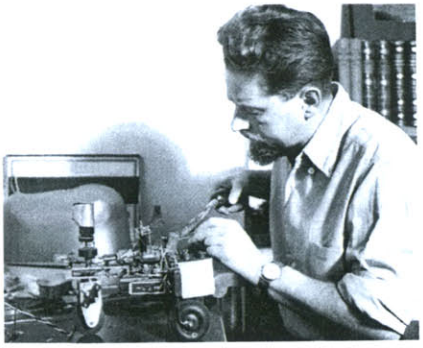


Figure 2.1: Grey Walter at work on one of his robots.

human brain presented to researchers in the early days of study.*[74]* He countered popular notions of the day that each piece of knowledge a being had was stored in the brain in individual units. Walter thought that the sheer number of units required, even if they were the size of neurons (on the order of microns in diameter), was far too large to be contained in the skull. Rather, he postulated that "richness of interconnection[s]" *[74, p118]* between neurons, nerves and the motor system allowed the complex behavior observed in animals to arise, and that these connections had been honed by eons of evolution to require minimal resources. From the humble amoeba, to insects, to reptiles and then mammals, Walter could see that something more than simple scaling of quantity of neurological matter was taking place that allowed these creatures to evolve increasingly complex behaviors over the millennia. For example, the honeybee (arguably a very simple creature) is able to convey the location of a food source that is miles away from its hive to within a few meters.*[23]* Not only can the honeybee find such sources with its limited sensory and processing capabilities, it can also return to its colony and direct others to the same spot. Effectively giving directions to another being is not an easy task, even for humans.

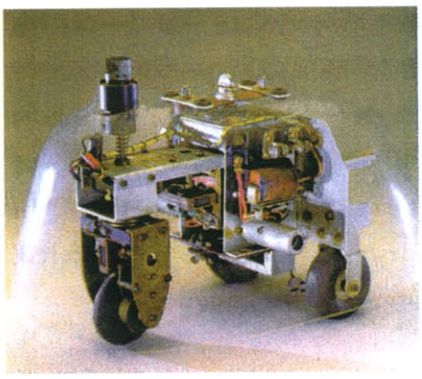


Figure 2.2: A reconstruction of Elsie, one of Walters' original "tortoises."

In an experiment to show that simple structures can produce surprisingly complex behaviors, Walter constructed two mobile robots, Elmer and Elsie, in 1951 (see Figure 2.2). These robots were intended to exhibit the simple animalistic behaviors of goal-seeking and scanning. To prove his theory that a minimal number of atomic units were necessary, given sufficient richness of connections between them, Walter used only two miniature vacuum tubes, two relays, two condensers, two small motors, and two batteries to construct them. The robots were also enabled with two "senses": light-sensitivity, given by a photo-electric cell, and touch, conveyed by a contact switch that closed when the shell of the robot encountered an

image courtesy [58], p14
image courtesy [75]

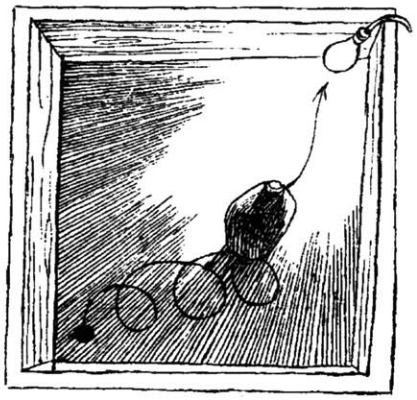


Figure 2.3: Drawing of one of the tortoises being attracted to the light source.

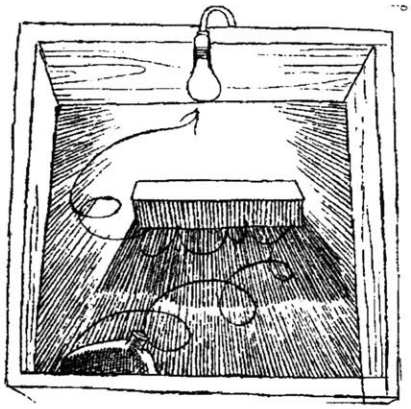


Figure 2.4: In the exploratory state in the darkness. The touch sensor causes the intricate movements, until the robot is finally free to pursue the light.

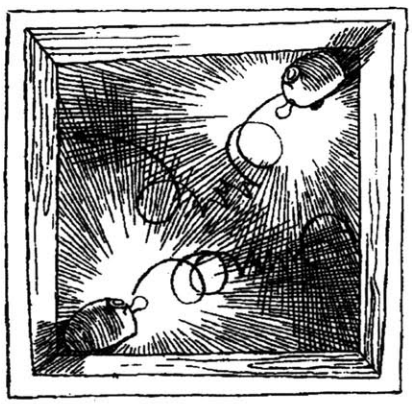


Figure 2.5: Two tortoises "dancing," each reacting to the other's headlamp.

images courtesy [75]

obstacle.

The circuits of *Machina Speculatrix*, as Walter named the species he had created, were designed to initiate exploration in darkness and to be attracted to areas of moderate brightness. Overly bright levels, on the other hand, were repulsive, and obstacles and inclines were unfavorable. With these simple traits prescribed, Elmer and Elsie were in fact capable of exploring their environment, shying away from both brilliant areas of light and corners of darkness, and either pushing around small obstacles or moving around large ones. *M. Speculatrix* was also observed exhibiting self-recognition and mutual recognition by means of head lamps that turned off when adequate light levels were reached. When placed in front of a mirror, or in each other's presence, the lamps would act as attractors, but would then turn off when the proximity was too close. Exploration would then be reactivated, at which point the lamp would turn back on, and the process would be repeated. See Figures 2.3, 2.4 and 2.5 for evidence.

One other behavior that emerged from the simple circuitry was that the robots would return to their hutch as their batteries ran down, in a mechanical analog of going to sleep. This occurred because as the batteries ran down, the photocell used to detect light registered lower values than before, so that very bright areas now fell within the range of attractive light readings. By design, the charging station for the robots was lit by a brilliant light bulb. Thus, as the environment was perceived to have lower light levels than it truly did, the bright light of the hutch became desirable, and once they entered, the robots would initiate charging of the batteries.

These robots were the first example of the reaction-based approach in action, and appeared to be on the right track to eliciting natural behaviors. There wasn't any complex "thinking" on the robots' part; rather they were

built intelligently such that computation was performed automatically, much the same way that animals seem to do. For example, when an ant comes across a twig for the first time, it doesn't have to know what it is, measure its entire size, and plan a path around it before moving. The ant doesn't need to know anything beyond approximately where the twig is relative to himself and needs only to guess which way he should start heading in order to get around it. The beauty of such simplicity was a compelling force behind the creation of Robotany. Unlike in Walter's work, however, a computer program is used to dictate the behaviors expressed as a function of the sensor inputs.

2.1.2 Vehicles that Love

Valentino Braitenberg is primarily a brain researcher, as was Walter, and used his knowledge of this field to write "*Vehicles: Experiments in Synthetic Psychology.*" Published in 1984, many years after Walter's *M. Speculatrix*, Braitenberg's monograph of thought experiments in behavioral psychology gave rise to an increase in experimentation in behavior-based robotics. Implementation of his hypothetical self-operating vehicles used only microchips and small motors, including those found in the LEGO™ Mindstorms kit◊. The simplicity of the physical and computational elements required to bring forth complex behaviors also inspired larger scale research efforts at the university level. He put forth the notion that direct connections between sensor input and motor output in varying manners can realize increasingly intricate behavior. When anthropomorphizing the behaviors exhibited by the machines, analogs of love, hatred, aggression, logic, foresight, and free will seemed to manifest themselves.

The first four types of vehicles put forth in Braitenberg's book each have sensors and motors. They vary in how the sensors are connected to the motors, both geographically

◊a Google search of the terms "Braitenberg" and "Lego" returns over 800 links representing numerous groups who have implemented these ideas themselves.

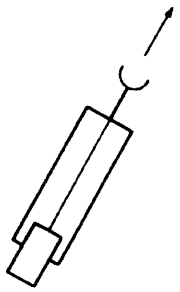


Figure 2.6: Braitenberg vehicle of type 1, with one sensor connected to one motor.

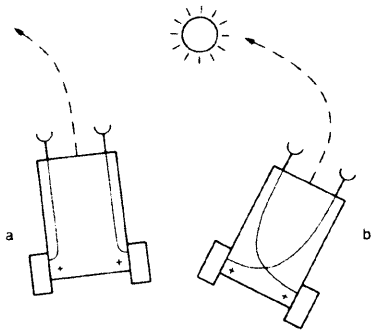


Figure 2.7: Braitenberg vehicles of type 2a (left) and Type 2b (right)

and mathematically. For instance, the first vehicle consists of one temperature sensor directly connected to one motor (See Figure 2.6). As this vehicle encounters regions of increasing or decreasing temperatures, it will speed up or slow down, accordingly. This is not a very interesting behavior in its own right, but is analogous in a simplistic sense to the thermostasis used by cold-blooded animals to moderate activity levels.

The next two types of vehicles each have two light sensors and two motors, located bilaterally and connected in a excitatory or inhibitory relation (See Figure 2.7). In a vehicle of type 2, there are two subtypes. Vehicle 2a, which has the left sensor connected to the left wheel and the right sensor connected to the right wheel. When these sensors sense more light, they cause the corresponding motor to spin faster. If a light source is located ahead of and to the left of center of the vehicle, the left wheel will rotate faster than the right one, executing a turn away from the light, where it will then slow down. If the light approaches again, the vehicle will repeat its escape maneuver, resting only when the light sensed is below its sensors' threshold. This vehicle has been termed a "coward" by Braitenberg, demonstrating its dislike of the light source by running away from it. In Vehicle 2b, however, the connections are crossed; the left sensor is now connected to the right wheel and the right sensor to the left wheel. Both connections are still excitatory, so that the more light sensed, the faster the motor will spin. Now, if a light is located ahead and to the left of this vehicle, the right wheel will rotate faster than the left one, affecting a turn toward the light. As the light intensity increases with decreasing separation, this vehicle will approach the light at increasing velocity, eventually smashing into it headlong. Braitenberg has termed this vehicle "aggressive," as it dislikes the light as much as its cousin 2a, but attacks the light, rather than hides from it.

In the type 3 vehicle, there are again two subtypes: Vehicle 3a

images courtesy [6], p.4 and 8

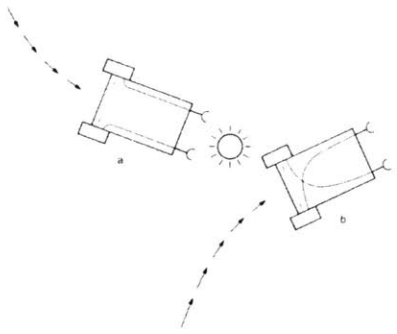


Figure 2.8: Braitenberg vehicle of Type 3a (left) and Type 3b (right)

with the straight connections (left to left, right to right) and Vehicle 3b with crossed connections between the sensors and motors (Figure 2.8). However, these vehicles differ from type 2 in that the connections are inhibitory rather than excitatory. In other words, when more of something is sensed, the motors will rotate more slowly in response. Again using light as the item being sensed, for Vehicle 3a, when a light source is located ahead and to the left of the vehicle, the left wheel will turn more slowly than the right, resulting in the vehicle turning toward the light and decreasing in velocity as it gets closer. The vehicle will initially race toward the light, eventually slowing to a stop before it in a head-on orientation, seemingly enraptured by its luminescence. This vehicle is expressing its "love" for the light source. Vehicle 3b also slows down in the vicinity of light, but prefers to keep its back to it. If the light were straight ahead of it, there would be no differential between the sensors, and it would also remain at rest facing the light. Any perturbation (such as would occur with real sensors), however, will cause the leeward motor to turn, inducing the vehicle to turn away from the light source until it has its back to it. This vehicle is termed an explorer by Braitenberg, one who likes the like source, but it always on the prowl for something better.

Braitenberg's work heavily influenced Robotany's core behavior, that of seeking out light and staying with it for the health of the houseplant. Robotany is connected in the manner of Vehicle 3a, the one that loves the light and comes to rest in areas of brightness.

2.1.3 Subsumption Architecture

In the mid 1980s Rodney Brooks brought attention to the field of behavior-based AI by being a vocal proponent of its advantages and how it contrasted the classicist approach (led by Marvin Minsky and Seymour Papert, also of MIT). He also began experimenting with his own robots using new

image courtesy [6], p.11

discoveries in the fields of cognitive science and biology as blueprints for the computer programs used to control them.*[12]* Stemming from their own dissatisfaction with the performance and complexity of available AI methods, several groups in addition to Brooks' set out around 1984 (shortly after Braitenberg's book was published) to find a different way to program robots such that they would be able to react in a timely manner to an unknown and dynamic environment around them, as creatures in nature would. While others worked on building AI systems that played videogames like a human *[1]* and eliminating the need for symbolism and representation *[61]*, Brooks led the group that developed situated and embodied mobile robots. These robots used real sensors in a real environment without internal world models or other *a priori* details.

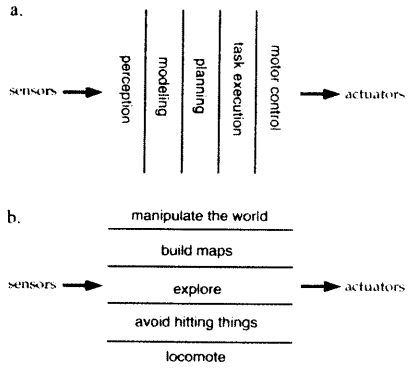


Figure 2.9: Control flow through classical (a) versus subsumption (b) architectures.

In his quest for something closer to nature, Brooks is credited for formalizing the notion of subsumption architecture, which directly contradicts the classical AI approach to robotics. In the subsumptive approach to intelligence, programs utilize a simultaneously computing stack of "layers" that each receive input from sensors and can affect actuators.*[13]* This differs from the classical methodology, which would step through a series of functions in time, from perception to planning to execution, a process which could take up to 15 minutes in some cases◊. Figure 2.9 graphically depicts the underlying architectural difference between the two methods of programming.

One advantage of the subsumption system is that layers can build upon each other. For example, in the figure above, the "explore" layer doesn't have to worry about obstacle avoidance, as this is taken care of by the routines in the "avoid" layer. In this manner, computation is distributed, leading to faster response times and thus more "natural" behavior. These routines were implemented by Brooks

◊One example is Shakey, a remote-controlled mobile robot built at Stanford Research Institute in 1970. Given carefully selected input data, it could plan and execute, in a period of several hours, paths to move from place to place and move objects around. See *[10]* for more information.

image courtesy [13], p.67

Figure 2.10: Genghis, a robot developed in the AI Lab of MIT.

in such robots as Genghis (Figure 2.10) and Atilla by way of AFSMs (Augmented Finite State Machines).*[12]* These AFSMs sent information from one to another in the form of a string of bits, to be decoded by the particular receiver for which the information was intended. This system had no central area of control, although in many cases an arbitration scheme was incorporated to deal with potential conflicts in motor output commands. The subsumption architecture also provided an avenue for redundancy, which improved robustness of the robots' outputs. If several redundant layers were processing sensor information in parallel, the resulting output was thus more reliable and complex goals could be achieved with relatively little computational power.

2.2 Obstacle Avoidance

Effective obstacle avoidance is critical to the survival of any mobile system. The ability to navigate an unknown environment to effectively reach the goal, whatever that may be, without getting stuck somewhere is the mark of success for a mobile robot. Many different methods exist to accomplish this task. One set uses pre-planned routes based on maps, computer vision or optical flow.*[5,14,48]* Another uses wall- or landmark-following, and others use primitive reaction-based methods such as bumping into a wall to trigger backing up and turning.*[18,39,44]* Although the preferred method is to navigate without making physical contact with the environment, when limited computational resources dictate strategy, anything that succeeds will suffice.

One of the most prevalent methods used for mobile-robot navigation, based on potential fields, is outlined below. A study of how humans navigate around obstacles on their way to reaching a goal, which provides a model for natural navigation, is also presented.

image courtesy http://www.ai.mit.edu/projects/genghis/genghis.html

2.2.1 *Using Potential Fields*

Applying potential fields to obstacle avoidance is a convenient technique that makes use of Newtonian dynamics in order to determine optimal navigation paths. By modeling goals and obstacles as regions of low and high potentials, respectively, one is able to determine a path to a goal that minimizes the work done. Potential fields have been used reliably by researchers for obstacle avoidance in mobile robots since the early 1980s.*[33,40]* The use of potential fields for navigation presupposes knowledge of the environment, which must be programmed into a computer that generates the robot's trajectory. Regardless of whether that computer is onboard the robot itself, or external to it, the path is commanded to the robot independent of *in situ* sensor readings. It is also possible for the robot to acquire active sensor data to verify the map. In order to do so, however, the inherently noisy nature of real sensors and dynamic environments must be taken into account.

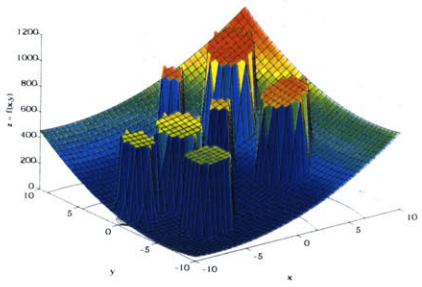


Figure 2.11: Representation of the obstacles as finite hills, the starting point as the top of a potential well, and the goal point as the bottom.

image from MATLAB simulation, [16]

For goal-seeking behaviors, a potential field is created in the shape of a paraboloid, or bowl, with the starting position of the robot set to be on the lip of the bowl, and the goal point set at the bottom. Obstacles are modeled as finitely high hills, with some gradient, which impart a repulsive force to the robot when it gets too near. Figure 2.11 depicts an example of a potential field used for goal-seeking. By simulating the physics of a ball rolling down the potential field along the path of least resistance, one is able to determine how the robot should navigate its planar environment. Specifically, the gradient of the potential field at a given point provides a force vector used to impel the vehicle toward its destination.

The following set of *[44,16]* provide a more rigorous mathematical description of potential fields. A goal has associated with it a potential, U_{goal} given by

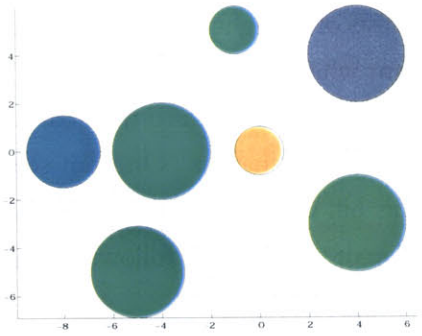


Figure 2.12: A set of circular obstacles in the field.

$$U_{goal} = \frac{1}{2}\xi * \rho(q) \ , \qquad (1)$$

where ξ is a constant, $\rho(q)$ is the distance from the goal, and q is the state vector $(x,y)^T$. The goal potential decreases as the distance between the robot and goal point decreases. As the robot tries to acheive a lower energy state, the goal therefore acts an attractor.

Obstacles, however, act to repel the robot with potential functions that increase with decreases distance. The obstacle's potential function, $U_{obstacle}$, is given by

$$U_{obstacle}(q) = \frac{1}{2}\eta\left(\frac{1}{th} - \frac{1}{\rho_o}\right)^2 \qquad for \quad \rho(q) \le th \quad (2)$$

$$= \frac{1}{2}\eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_o}\right)^2 \qquad for \quad th < \rho(q) \le \rho_o$$

$$= 0 \qquad for \quad \rho_o < \rho(q)$$

where η is a constant, ρ_o is the radius of the obstacle, modeled as a circle, and $\rho(q)$ now represents the distance from the obstacle. Since the robot is modeled as a point mass, the parameter *th* has been added to give the minimum distance between an obstacle and the robot. This term serves to limit the range over which the obstacles can affect the path of the robot and makes the function bounded when $\rho(q)$ is equal to zero.

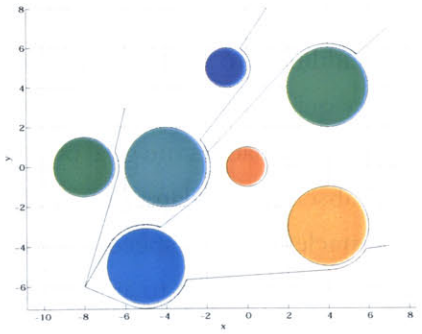


Figure 2.13: Paths through the obstacle field to the goal point.

images from MATLAB simulation, [16]

The total potential function for this system is then given by

$$U_{total} = U_{goal} + \sum_{i=1}^{\#\,obstacles} U_{obstacle\,i} \qquad (3)$$

where U_{goal} and $U_{obstacle}$ are defined in *(1)* and *(2)* and summed for all obstacles in the environment. To determine the trajectory for the robot, one takes the negative gradient of this energy equation given above to obtain the force exerted by the potential field on the robot at any given coordinate along the path q. This field acts to impart a virtual gravitational effect on the robot, akin to a ball rolling down a hill, which directs it toward the goal on the most efficient, path.

The path taken by a robot consists of straight sections towards the goal interrupted by sections which follow the contour of an obstacle in its way, until a line-of-sight path to the goal is restored. Figure 2.13 illustrates this behavior. One can see that, as a ball rolling down this hill would, the robot traces a straight-line path up to the edge of an obstacle, follows along its edge, and continues on its way to the goal.

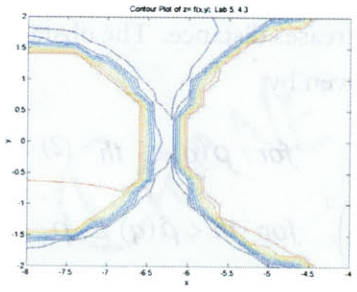


Figure 2.14: Overlapping potential fields, which prevent the robot from passing between these obstacles.

image from MATLAB simulation, [16]

One disadvantage to the potential fields method is that it is very easy for the robot to become stuck in a convex space. Figure 2.14 shows an instance of this where two obstacles are located close together. The gradient surrounding these obstacles, which acts to keep the robot a safe distance away from them, creates an overlapping region of high potential that the robot cannot overcome. If the robot's path brings it to this point, it has no recourse to get out of its potential well and is stuck for perpetuity.

Another disadvantage to the potential fields formulation is the lack of damping in the system, causing the solution to be jerky and discontinuous. This can lead to "unnatural" motions, where the robot has to pause at obstacle boundaries and change orientation before continuing along. Such sharp turns may be impossible to execute if the vehicle happens to be non-holonomic, but there is no recourse in the algorithm to account for this fact. Also, if the algorithm is used to determine the robot's speed in addition to the direction of motion, the robot will move very quickly when it is far from the goal point, and very slowly as it approaches the goal point. This fact combined with the absence of damping could lead the robot to crash into obstacles at high speed. Some researchers have added damping to the equation to ameliorate these effects to a degree.*[33]*

The main disadvantage of this method for any autonomous robot is that it relies so heavily on maps and calculations

performed before the robot can start. If the robot can accurately map its surroundings, it will have a chance at succeeding in reaching its goal (barring complications such as local minima). However, to do this in a "natural" fashion, i.e. without stopping for long periods of time while it calculates its next move, or changing direction abruptly, the robot would require a significant amount of processing resources to plan and enact the desired path in a timely manner.

Instead of gravitational potential fields, some have tried creating virtual electrostatic potentials to guide their robots through two-dimensional environments.*[72]* In this method, a virtual resistor network is created to represent the environment, from which the laws of electrostatic fields are used to solve for an efficient path through the obstacles to the goal. Rather than using a map of the environment given before navigation, the robot builds a map, and therefore its resistor network, via sonar readings. In this experiment, the system lays a grid over the environment, and then sensor readings determine whether a particular cell is occupied with an obstacle or not. These obstacles are then assigned resistances, with densely occupied areas assigned higher resistances. Then using Gauss' laws, a unique solution for a path of least resistance through the closed resistor network allows the robot to follow a nearly optimal path through the environment.

The use of potential field methods, either gravitational or electrostatic, has resulted in successful navigation in both simulation and actual tests on a mobile robot. However, the process is computationally expensive and slower than "natural" responses to the environment would be. Also, the tactic of breaking the environment up into regular units or cells and the creation of a complete map of the environment before navigation can be attempted seem contrary to nature's way of working.

2.2.2 *Behavioral Dynamics of Steering*

When thinking about navigation of mobile robots, it is helpful to take a look at nature's approach to the matter. As efficient solutions have been evolved over the millenia, they can provide insight into how robots might be able to use these techniques as well. In their experiments, Fajen and Warren model the behavioral dynamics of steering in humans to show that explicit path planning is not necessary.*[24,25]* They accurately simulate natural behavior with an empirically determined second-order model inspired by measured data of human navigation. This model is a function of the angles and distances (relative to the navigator) of goals and obstacles, which act as attractors and repellors, respectively.

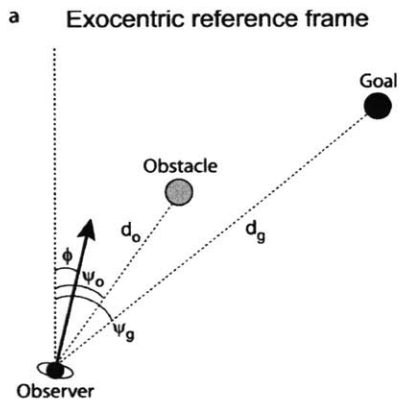


Figure 2.15: Coordinate system used in the equation of dynamics

Given one goal and one obstacle, Fajen and Warren's model is analogous to a spring-mass-damper system and takes the form

$$\ddot{\phi} = -b\dot{\phi} - k_g(\phi - \psi_g)(e^{-c_1 d_g} + c_2) + k_o(\phi - \psi_o)(e^{-c_3|\phi - \psi_o|})(e^{-c_4 d_o}) \quad (4)$$

where ϕ is the heading angle measured in a fixed reference frame, the ψ_g and ψ_o terms are the angles of the goal and obstacle (also in the exocentric frame), and the d_g and d_o terms are their radial distances from the navigator. The c terms are parameter gains that modulate the strength of response to goals and obstacles and the k terms represent effective "spring constants" associated with each goal and obstacle.

The first term on the right-hand side of equation 4 acts as a frictional ("damping") force that opposes angular motion. Next there is a "spring" term which pulls the navigator toward a goal. This is modulated by an exponential so that attractiveness decreases with goal distance, yet is scaled by c_2 so that acceleration never reaches zero. The third term represents the effect of the obstacle and is also modeled as a spring force. This time, however, the spring pushes the

image courtesy [24,25]

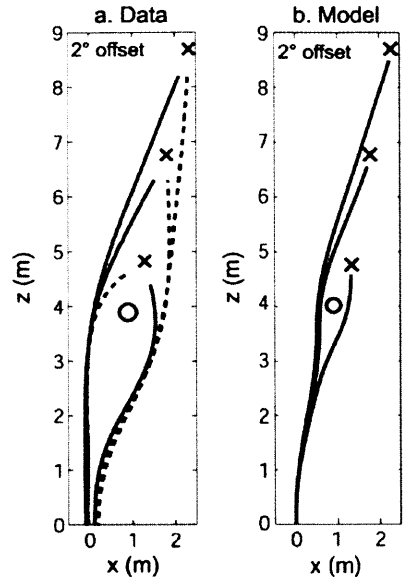


Figure 2.16: Path taken by human around an obstacle as obseverd by Warren et al in a virtual environment(l) and the path taken by the model under the same circumstances.

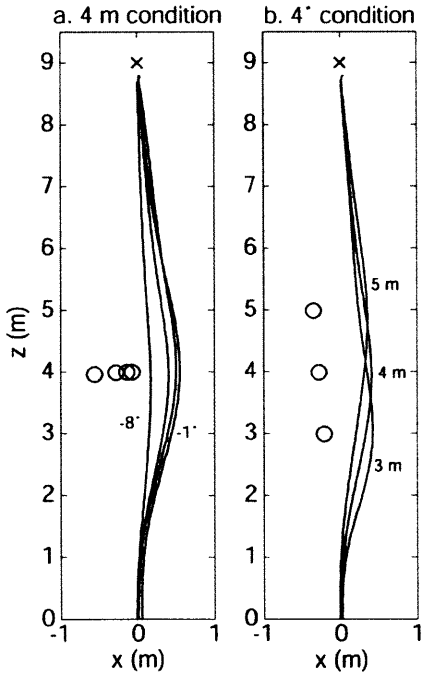


Figure 2.17: Path generated by the algorithm in (4) for a range of goal and obstacle conditions.

images courtesy [24,25]

navigator away from its desired path and is modulated by an exponential with an asymptote at zero. This asymptote serves to push the navigator away from the obstacle in the proper direction and at a high rate when its approach is nearly head-on. The effect of the obstacle, like that of the goal, also decreases with increasing distance from the navigator. It is interesting to note that the transverse speed of navigation is not a variable in the steering dynamics, as it was nearly constant in all of the human navigation data upon which this model is based.

An interesting fact that arises from this model is that the terms for goals and obstacles can be combined linearly, suggesting that it may be a contracting system. The rates of convergence to a goal are equal for various initial goal angles, as would be expected from a contracting system. The superposition also implies that the behavior does not get complicated in greater than linear fashion with the addition of obstacles to the environment. Contracting systems are discussed further in section 2.4.

The salient features of Fajen and Warren's model show that human navigation is not as complex as previous models have implied. That the effects of obstacles decay exponentially to negligible levels indicates that humans only need to sample the next few objects in their environment pertinent to their current path, rather than trying to map the entire scene before starting out. Also, the lingering effects of goals and obstacles in the model shows that humans (and therefore their simulated counterparts) remember goal forces. The final route chosen arises from a natural competition between goal attraction and obstacle repulsion in the behavioral dynamics. The act of turning in this model is not a set biomechanical process but is instead governed by the navigator's movement relative to the objects in the scene. Perhaps the greatest simplification found is that nowhere in the model do system dynamics

– or self-knowledge of how the act of turning is carried out – come into play.

2.3 Homeostasis

Homeostasis is a basic biological process used by all living creatures to maintain balance and harmony between the internal functions of the body and the external effects of the environment.*[2]* For example, when the weather is warm, humans perspire to aid cooling. When a meal that is high in sugar has been consumed, the pancreas produces more insulin to compensate. In most animals, the endocrine system performs homeostasic functions necessary for survival on a level far below any conscious control. The fact that the proper chemical conditions can be maintained without input from the central nervous system has made homeostasis has been a critical development in the saga that is evolution. By decreasing the computational resources required for basic functions, homeostasis effectively frees up the brain for higher-level processes, such as art and abstract thought.

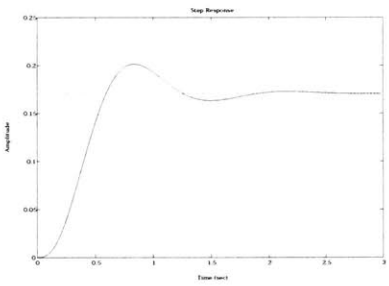


Figure 2.18: Stabilizing a system via a negative feedback loop

A homeostatic system maintains its steady-state values by way of negative feedback control to stabilize whatever levels are being measured when they are disturbed. Akin to classical controls systems, the endocrine system attempts to restore balance by releasing specific amounts of chemicals into the bloodstream based on how far monitored levels have deviated from the norm (see Figure 2.18).*[35]* In the same respect, homeostatic systems do not act as switches to begin or halt processes, but more like dimmer knobs, altering the rates at which these processes occur. For instance, horomones are constantly emitted by the various glands of the body, albeit in baseline quantities in periods of low stress. However, when danger is sensed, before any conscious thoughts or plans are formed to cope, the body releases hormones that increase respiratory and heart rates, tenses the muscles in preparation for physical exertion, halts digestive processes to save energy resources, and causes the liver to release sugars, fatty acids

and cholesterol into the bloodstream to provide energy. Similarly, when the danger has abated, the endocrine system normalizes the processes that were affected by reinstating pre-attack levels of the body chemistry. If this weren't an automatic reaction, and creatures had to mentally prepare to avoid being eaten, for example, evolution would have likely halted long ago.

For Robotany, homeostasis comes into play by monitoring internal plant variables, such as amount of light received and humidity of the soil, and external variables such as ambient temperature. Depending on these values, the characteristics of its behavior will change. The homeostasis routine is not a subroutine of higher-order functions. Rather, the routine updates certain biases in the system based on its readings, and these biases affect the strength of response to certain inputs. For instance, if the plant has been unable to reach sunlight, or naturally has a very strong desire for light, it will be willing to get closer to obstacles in order to get more sunlight (to within a hard limit determined by the physical dimensions of its chassis). On the other hand, exposure to excessive heat will reduce the plant's desire for sunlight and increase its sensitivity to light levels, making darker areas more attractive as a goal placement.

In a similar manner, Grey Walter's Elmer and Elsie exhibited this same sort of balance between desire for moderate light levels and aversion to overly bright areas, as discussed in section 2.1.1. In addition, as their internal measurement of battery levels decreased, sensitivity to the readings decreased and brightly lit areas fell within the range of acceptable levels for the robots. This created the behavior of sleeping, as they would be attracted to their brilliantly lit charging stations. Simple homeostatic relations were therefore as essential to the survival of htese robotic creatures as they are to biological creatures.

Figure 2.19: Walter Cannon

image courtesy http://www.harvardsquarelibrary.org/unitarians/cannon_walter.html

2.3.1 The Grandfather of Homeostasis

Walter Cannon was a neurologist and physiologist in the early 20th century credited with, among other things, the processes behind homeostasis and how they affected the body. Although his background was in the biological sciences, and his path to homeostasis was through study of the human body, he also applied the principle to politics and society as a whole. The body's ability to stabilize itself and to prepare for crisis was such a powerful notion for Cannon that he could see how it could benefit other complex systems as well.*[15]*

Cannon drew analogies between external influences on the internal state of the human body and the tribulations faced by social groups – such as families, industries, and governments. He suggested that these social structures could benefit from applying the tenets of homeostasis to ensure their survival, just as the human body benefits from its endocrine system. He drew many parallels between "the body physiologic" and "the body politic," including comparing blood, which circulates the hormones necessary for the organs to change their behavior, to money, which replenishes the depletion of goods in the market. Just as the brain is freed from drudgery to perform higher-order functions, such as dance and poetry, society can similarly be freed and allowed to explore its full potential.

Although he may not have foreseen the application of homeostasis to the field of robotics, Cannon could see that it had a much longer reach than solely the living body. The underlying principles that maintain stability in the face of radical circumstances can guide the formation of varied complex systems into becoming coherent structures capable of greater things.

2.3.2 *The Many Whims of Kismet*

The interactive robot Kismet was created by Cynthia Breazeal, a professor whose research focuses on human-robot interaction. Kismet was designed to evoke a caretaker-type behavior from the humans that interacted with it. To do this, it was programmed to use facial expressions to convey emotional states that would come up in adult/infant interactions; emotions such as happiness, sadness, boredom or sleepiness (see Figure 2.20).

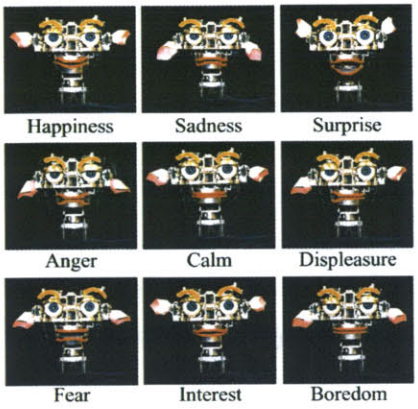


Figure 2.20: Kismet's range of emotion and expression.

image courtesy [8]

In many ways, Kismet was an experiment in homeostatic control applied to human-robot interaction. Its fundamental intelligent capability – being able to transition between emotions in a natural manner that would keep the interest of the human participant – was acheived using a homeostatic scheme. This scheme served to maintain an internal balance of behavioral drives such as sociability, stimulation, security and fatigue.*[8]* Kismet was able to keep track of these drive levels using a set of internal variables, such as time spent in interaction, and external variables, such as volume and pitch of audio input or distance of a human face from its own.

When Kismet sensed that its various drive levels had gone out of homeostatic range, it displayed signs of anger or boredom. This encouraged the human participants to help it restore balance by calming it or engaging it, as one would do with an infant. If the human failed to do this, or made Kismet more upset, it put itself to sleep, almost as a defense mechanism, in an effort to adjust its homeostatic levels itself.

All of Kismet's behaviors were modeled after those observed in human infants by psychologists and behavioral scientists. For example, if no one had spent enough time with it recently, Kismet would start making noises to draw attention to itself, just as a baby cries to get attention. If, however, it felt "over-stimulated" (i.e. too much noise or contact), or

had been interacting with someone for an extended period, it would withdraw from the interaction and act tired to let the participant know that it didn't want to play anymore.[7]

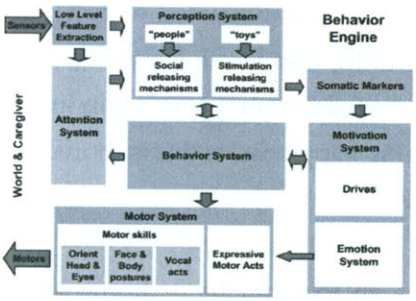


Figure 2.21: The control flow of Kismet's program that regulates behaviors as a function of external inputs.

Kismet not only reacted to its environment internally, but also gave information about its current state in an outward manner. Kismet was able to convey its internal state to human participant in a familiar visual manner by adjusting its facial expressions. Several levels of increasingly complex motor control were required to produce these expressions. The first is a set of motor primitives that set the range of motion and how quickly to transition between positions. Next is a skill level that takes care of moving several motors in a coordinated fashion, such as raising the eyebrows, or wiggling the ears. The uppermost level combines the various skills into cohesive expressions to convey meaning to observers that accurately reflect Kismet's internal emotional state. Changes between states are blended by averaging the various emotions that are being evoked.[7]

Kismet is an example of a complex autonomous robot that is enabled with a rich range of emotion, as well as the ability to display this range while interacting with human participants. Unfortunately, to be able to achieve such richness, many computers were occupied full time with such things as vision processing, auditory processing, motor control, and implementation of the homeostatic system that arbitrated between emotions. The fact that it was able to achieve such success in its goal was inspiring, however, to note that homeostasis was an effective method to rouse natural responses from caretakers who forgot that Kismet was just a robot.

image courtesy [8]

2.4 Contraction Theory

Contraction theory is an analysis technique used to gather information about the stability and controllability of nonlinear systems.*[47]* This technique uses state space representations and stems from sliding control theory as introduced in *[65]*. It is able to simplify analysis of high-order systems of dynamics by introducing new variables that give more information about the controllability of the system at hand.

For a system to be contracting within a certain region means that any trajectory s_i starting in that region will converge to some goal trajectory, s_0 within the same region. This implies that the final state of the system is independent of initial conditions. Two important traits of a contracting system are: a superposition of contracting systems is in itself contracting, and if a system is contracting in any point in time, it is contracting for all time. The latter notion is particularly useful when determining the stability of following a moving goal point or avoiding moving obstacles. A system that is contracting in all regions of the state space is said to be globally contracting.

Contraction analysis has proven to be particularly helpful when analyzing position control of second-order dynamic systems.*[32]* Consider a given second-order system

$$\ddot{x} = f(x,t) + u(x,t) \qquad (5)$$

where t is time, x is the position state vector, $\ddot{x}$ is its second time derivative, $f(x,t)$ governs the dynamics of the system, and $u(x,t)$ is a feedback control signal. The problem of controlling position in a force field can be transformed into one controlling position in a velocity field via the introduction of the variable

$$s = x + T\dot{x} \qquad (6)$$

where x is still the state vector, $\dot{x}$ is the first time derivative of this state vector and T can be thought of a damping coefficient of sorts that modulates how closely s predicts the x trajectory T seconds later. The second-order equation above, *(5)*, can now be

tranformed to a first-order system in s,

$$\dot{s} = f(s,t) \qquad (7)$$

where $\dot{s}$ is the first time derivative of s. The trajectories of this first-order system can be visualized as stream lines in the velocity field $f(s,t)$.

Using the simplified description of the system given by *(7)*, one proceeds with the contraction analysis by evaluating the Jacobian, or partial derivative with respect to s, of the system $f(s,t)$. If the eigenvalues of the symmetric part of the Jacobian are strictly less than zero, then the system is contracting. This can also be written as

$$\frac{1}{2}\left(\frac{\partial f}{\partial s} + \frac{\partial f}{\partial s}^{T}\right) \leq -\beta I < 0 \qquad (8)$$

where β is some positive number, I is the identity matrix of the same dimension as the Jacobian, and ^{T} denotes the transpose operation. The graphical result of this statement is that all stream lines in the velocity field converge to a single trajectory, like a laminar fluid flow compressing through a nozzle.

Finding that s is contracting in its velocity field implies that x is also contracting in its force field. This is due to the definition of s as a first-order system of x and $\dot{x}$ in *(6)*. The solution of a first-order equation implies exponential convergence to a particular solution. Thus, since s converges to a particular solution of the velocity field by contraction, x must converge to the particular trajectory specified by s.

Since the resulting trajectory is unique, regardless of initial state, any linear combination of such fields can also be shown to be uniques as well, and therefore contracting. This conclusion agrees with biological experiments described by Bizzi and Mussa-Ivaldi regarding motor primitives in frogs.*[56]* Motor primitives are unique trajectories of the limb elicited by stimulation of a particular section of the frog's spinal cord. When two such sections were stimulated, the resulting motion appeared to be a

linear combination of the individual trajectories. This finding supports the mathematics behind contraction theory by showing that nature perhaps also uses a linear combination of simple motions to create complex behaviors.

The description of contracting behavior thus far applies only to attractive points in the trajectory space, such as goals. For repulsive force fields that may be associated with obstacles in the terrain for example, there is no singular solution. The same definition for s is used in this case, as well as the same control function, but the velocity field associated with the obstacle is on the order of $1/r$, where r is the distance from the robot to the obstacle, with the zero point (corresponding to infinite repulsive force) located at the center of the obstacle. As can be seen intuitively, any given set of initial conditions will result in different trajectories. In fact, all trajectories radiate away from the obstacle linearly. When plotted in polar coordinates, however, these trajectories are straight, parallel lines. They are not converging, but they are not diverging either. Haag and Slotine supposed that if this repulsive field were added to a strongly contracting one, that the result would still be contracting. This did indeed prove to be the case mathematically, and a new lens to study obstacle avoidance through was developed.

Using contraction theory for obstacle avoidance does not necessarily free us from the problem of local minima as found with potential fields, however. That problem can only be eliminated if the entire region containing the trajectory is contracting, due to the fact that by definition a contracting system can have only one solution in the contracting domain. If the trajectory starts in the contracting domain within a certain radius tof the attractive point, it will find its way to that goal. If however, the trajectory leaves the contracting domain, or starts outside that ball of influence, it cannot be guaranteed that the trajectory will not become mired at a local minima point.

2.4.1 *Contraction of Steering Dynamics*

Section 2.2.2 described a dynamical model of human steering, introduced by Fajen and Warren, that is stable and efficient, as well as modeled on naturally occurring behavior. It is of interest to examine the properties of this model because evidence of contraction in a natural system would strengthen the validity of using the approach in other areanas.

When analyzed with the techniques outlined in the previous section, however, the dynamics are found not to be globally contracting. The first step in analysis was to linearize the equation as it was given in section 2.2.2, and repeated below.

$$\ddot{\phi} = -b\dot{\phi} - k_g(\phi - \psi_g)(e^{-c_1 d_g} + c_2) + k_o(\phi - \psi_o)(e^{-c_3|\phi - \psi_o|})(e^{-c_4 d_o}) \quad (4)$$

Since the distances and angles of goals (d_g and ψ_g) and obstacles(d_o and ψ_o) are not constant but vary with time as a function of heading, the exponential terms make the system nonlinear. This can be remedied by noting that since the constants, distance values, and the absolute value of angle difference are always positive, the exponentials are bound between zero and one. These terms can then be replaced by constant that vary in the same range. In this way, the dynamic equation can be reduced to

$$\ddot{\phi} = -b\dot{\phi} + (\alpha_2 - \alpha_1)\phi + C \quad (9)$$

Where α_1 is bounded by 3.0 and 10.5 and α_2 is bounded by 0.0 and 198.0. Now that the linear equation is found, the Jacobian can be computed, giving a square matrix equal to

$$J = \begin{bmatrix} 0 & 1 \\ \alpha_3 & -b \end{bmatrix} \quad (10)$$

where α_3 is the sum of α_1 and α_2, resulting in a range of -10.5 to 195. According to the theory put forth in *[44]*, contraction is determined if the eigenvalues of the symmetric part of the Jacobian are strictly less than zero. The symmetric part of the Jacobian of the dynamics is given as

$$J_{symm} = \begin{bmatrix} 0 & \frac{1}{2}(1+\alpha_3) \\ \frac{1}{2}(1+\alpha_3) & -b \end{bmatrix} \qquad (11)$$

which results in eigenvalues of

$$\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 + (1+\alpha_3)^2}}{2} \qquad (12)$$

One can see that the pair of eigenvalues will always have one positive member, regardless the value of α_3. This means that the system is not contracting, at least not in the identity metric.

The above calculation can also be performed while retaining the nonlinearities. It becomes necessary to introduce the composite variable

$$s = x + T\dot{x} \qquad (6)$$

where x is the state vector of ϕ and $\dot{\phi}$ to produce manageable results. However, the same Jacobian resulted from the process, which indicates both that the linearized approximation was valid and that the system is in fact not contracting. Even if one analyzes solely the attractive part of the dynamics given by *(4)*, one of the eigenvalues will still be positive, independent of the values of the variable terms. This tells us that even without the addition of a diverging field associated with an obstacle, the system is not contracting mathematically.

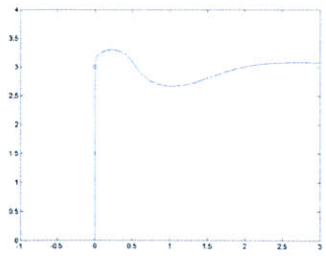


Figure 2.22: The effect of the obstacle vanishes when its angle relative to the navigator is equal to zero.

To test the validity of the results produced by Fajen et al., a MATLAB simulation was programmed using the full dynamics as given in the appendix of *[25]*. These simulations did indeed give the same results as presented in their paper, albeit for a narrow range of goal and obstacle angles. Figures produced by this simulation show that the initial condition is unimportant within a given region. This holds with the conclusions of contraction theory, so the fact that the mathematics don't comply suggests that there may be some mathematical approximation of the true dynamics that does contract.

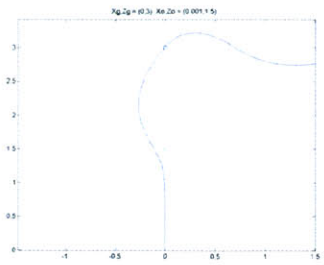


Figure 2.23: When the obstacle is moved by even 1/1000th of a unit, the system dynamics cause the navigator to avoid it.

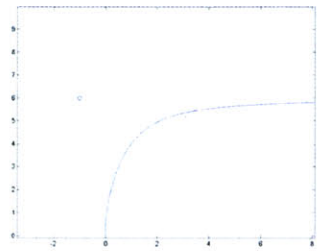


Figure 2.24: The goal point is located at a negative angle relative to the observer, causing the dynamics to treat it as a repulsive point.

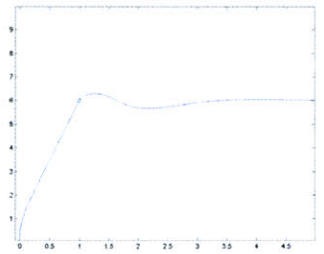


Figure 2.25: Response predicted by Fajen et al.'s dynamic equation.

One guess at why the system equation doesn't contract is that since Fajen and Warren's model of steering was determined empirically rather than from first principles, the mathematical foundation is not strong enough to explain why paths result the way that they do. This assumption is strengthened by the simulation of the case where one obstacle is located directly ahead of the navigator and one obstacle located directly in the path between the navigator and its goal. In this situation, shown in Figure 2.22, the path of the observer approaches the goal directly, passing straight through the obstacle. This is because when the angle of the observer is directly in line with the obstacle, the repelling term vanishes, due to the (ϕ-ψ_o) factor of *(4),* which is the angle of hte goal relative to the navigator. If the obstacle is moved to either side of the y-axis by 0.001, as seen in Figure 2.23, the path changes and the obstacle is effectively avoided.

Another example where the simulation doesn't hold is when the goal angle is negative and the obstacle is out of range of influence. In this case, the path never approaches the goal directly. Instead it goes straight ahead and then always veers away to the right to continue out to infinity. (Figure 2.24) It would seem from intuition that this asymmetry should not occur. However, the equation of dynamics shows that as the sign of the difference between the navigator angle and the goal angle changes from positive to negative, the resulting force changes sign as well. It then behaves in the same way as the repulsive force associated with obstacles. This shift does not occur, however, when obstacles are located at a negative relative angle to the observer. Why this dichotomy was not resolved in the equation is unknown, but it explains why all the figures in Fajen and Warren's paper have the goals located ahead or to the right of the observer's start point.

2.5 Cooperation and Competition

Whenever multiple autonomous agents operate in overlapping physical regions, they will likely feel the results of each others' presence. If they need to cooperate on a given task, there must be some way for them to share information. There are many possible communiation schemes that could be implemented, ranging from directed to implicit methods. Directed communication relies on explicit recognition of agents in the community, either by the agents themselves or an omniscient agent that can communicate with all of the others. Implicit communication, on the other hand, occurs when the robots coexist and happen to cooperate and/or compete due to having similar goals.*[4]*

Many applications for multi-robot teams exist that require explicit communication methods. The first of these is minesweeping applications where many robots work together to ensure complete coverage of terrain.*[30]* Minesweeping robots build upon the technology used in surveillance and reconnaissance robots that function as distributed systems. By distributing the work with information-sharing schemes, a team of robots can thoroughly investigate a given area in much less time that an individual robot.*[57]* A set of manufacturing robots, each of which may have a unique task, often need to communicate information regarding these individula tasks to each other to ensure a cohesive result.*[29]* One of the most common testing grounds of this research is the annual RoboCup competition.◊ In this contest, teams from around the world strive to create the most effective robotic soccer team and often create new techniques in multi-robot interaction in the process.

While cooperation may often require having multiple robots work together to fulfill some higher-order goal as in the examples above, competition can arise in a much simpler situation, in which multiple robots are functioning in a shared physical space and have similar goals.*[57]* Each robot strives to complete a personal task while constrained by limited resources. Those robots with

◊RoboCup competitions have turned into effective proving grounds for the projects developed by many research groups. At last year's competition, over 200 teams vied for the title of champion of their division. See *http://www.robocup.org* for more information on the contest.

the more efficient routines will be more likely to complete their tasks. Examples of this include foraging robots and those that need to consume resources from the environment. When robotic systems that exhibit competitive impulses are tested in simulated environments, individuals can often reach deadlock and can be immobilized in their efforts. Fortunately, in the real world, sensor and actuator uncertainty act to prevent the problem of perfectly equal balances from occurring and most conflicts can be resolved.

The following section discusses research that deals with multi-robot interactions that can result from behavior-based approaches.

2.5.1 Robots that Flock and Forage

It is often found in nature that basic actions of individuals can produce complex behaviors when observed in an aggregate form. Similar occurences can be observed in groups of autonomous agents as well, as seen in the work by Matarić *[50,51]* and Arkin *[2]*.

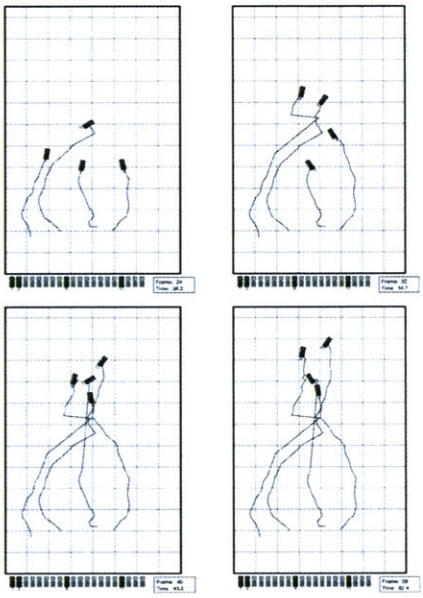


Figure 2.26: Recorded data from Matarić's experiment on flocking.

image courtesy [51]

In her Ph.D. thesis*[51]*, Matarić explored different combinations and configurations of individual robots. With several basic behaviors enabled on each robot, groups of individuals acting in the same vicinity can give rise to larger scale behaviors. For instance, by combining avoidance, aggregation, and wandering behaviors, flocking was observed in one group of individuals. Incorporating the homing behavior further allowed the flock to direct itself towards a particular goal location as a single entity. Figure 2.26 shows the results of several of these experiments recorded in the physical environment. These results were observed in the simulated implementation as well.

Matarić's robots have also performed cooperative tasks, such

as foraging, by way of stigmergic interaction.[52] Stigmergy involves interaction through indirect manipulation of the environment. One example of this in nature is how animals mark their territory with urine or musk, to make other animals aware of their presence and strength. The robots can also communicate indirectly by broadcasting their current state to others within range, like elephants trumpeting to declare their territory.

Figure 2.27: A group of Khepera robots used in several of Matarić's experiments, including those on forming flexible, adaptable robot formations.

image courtesy [53]

The foraging test put forth by Matarić for her robots involved having them seek out metal pucks and bring them to an unspecified gathering location. This function worked as planned, both in simulation and in reality. In one trial of the physical implementation however, an unexpected result occurred when the researcher put down a plate to act as a model for a clump of pucks to catalyze the process. Rather than bringing all the pucks to the plate, the robots were found to be pushing the plate to be on top of the pile of pucks that they had made. This is an excellent example of the inadequacies of simulation when trying to develop solutions for use on physical robots in the real world. The various unknowns in effect in the physical implementation can never be modeled exactly, which means that the future states cannot be predicted in simulation. Therefore explicit planning is unrealistic and not feasible as a means of developing complex behaviors. This fact also makes it more difficult to create complex behaviors by design, as the end result is never certain in all circumstances.

Balch and Arkin [4] point out that another benefit of the behavior-based approach to multi-robot systems is that it results in a distributed system, without a central planner or arbiter. This provides a natural immunity to individual robot failures and inaccuracies, since a centralized system could interrupt the success of the group as a whole. [4] The robots acting as individuals could lead to slower times to complete

group tasks, but sharing information between robots can help to alleviate this problem.

Later work by Matarić introduced arbitration schemes based on dominance hierarchy, caste differences, or territorial claims. These arbitration schemes aimed to reduce the interference that is inherent in large group of individuals, each with common goals to achieve. Decisions were usually implemented by way of direct sensing or else explicit communication. Research by Arkin et al. shows that such communication of behavior and goal states to other robots saves time of task completion and increases efficiency.

2.6 Localization and Mapping

In addition to being able to navigate various environments, biological creatures are also able to know where they are, where they're going, and most importantly, where they came from. To overcome a lack of memory capabilities, many creatures have devised ingenious methods to find their way home. For instance, ants leave trails of pheromones leading from the home to food sources, which can be followed back to return safely.*[43]* Honeybees convey locations of pollen to their hive-mates by means of a dance that corresponds to distances and directions traveled from the hive and back.*[23]* Additionally, some creatures have redundant means for navigating and returning to their origin point. For instance, homing pigeons have been found to use as many as four different methods, including vision, magnetic fields, polarization of sunlight through the atmosphere, and stellar maps.*[77]* A robot based on biological systems should also be able to incorporate several different successful methods for mapping.

When applied to robots, localization and mapping are heavily dependant upon the reliability of sensors used. Building rigorous maps of the surroundings requires the use of not only odometry

(which is inherently unreliable due to slipping) but also landmark detection and recognition. This is in addition to being able to discern information about the robot's location relative to local objects. The ability to retain these maps in memory can be a challenge for robots with limited resources, such as the Robotany vehicles. In such situations, novel approaches must be used to overcome these limitations. Instead of remembering where it came from, the robots could simply home in on a beacon at their origin. If they could switch between following light and using the beacon at the proper time, building a map would be unnecessary.

In order for a map to be of any use, a robot must know where it is located in relation to the rest of the environment. This task, known as localization, is usually performed by corroborating sensor data with odometry readings verified by the robot's control system. Localization can be achieved with or without *a priori* knowledge, although the latter requires complicated statistical mechanisms such as Bayesian networks in conjunction with Kalman filters.*[68]* While such statistical methods may provide many advantages in an idealized setting, they have been found to be susceptible to errors and drift from the sensors. Some of this error can be ameliorated by having several robots collaborate and verify data between themselves to arrive at a more robust approximation of the scenery.

2.6.1 SLAM

Many robots have been programmed with the ability to follow maps already in their memories, or to localize themselves in relation to known data points, but truly autonomous robots must be able to function without any such knowledge. The technique of Simultaneous Localization and Mapping, also known as SLAM, is designed to do just this.*[20,55,68]* When a robot is turned on in an arbitrary location and orientation, this scheme allows it to infer its current location and to build an internal map with reference to its sensor measurements.

As the robot takes in sensor data, it concurrently updates its perception of the environment as well as its own location within that setting. These two points of data reinforce then each other to provide a more accurate representation. As the robot travels around the available area, it updates its map of landmarks and self-location with the increasing amount of data, serving to concretize the burgeoning internal image.

Traditional SLAM techniques, such as that put forth by Thrun *[68]* and Williams *[78]*, rely on Kalman filters in conjunction with sequential Monte Carlo methods (acting as particle filters), which increase in complexity quadratically with the addition of more landmarks to the environment. This is because as new data about any one landmark is acquired, everything known about all of the other landmarks must be updated as well. The super linear complexity of traditional SLAM techniques do not scale well as the number of landmarks increases to greater than 500 or so. Unfortunately, the real world can easily contain upwards of one million landmarks.

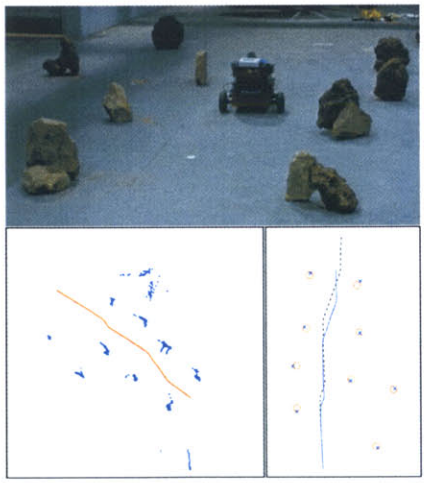


Figure 2.28: The results of a mapping task using SLAM. (top) The robot and obstacles. (bottom) The raw data of perceived locations of obstacles on the left, and obstacles localized with SLAM (dots) and manually (circles) on the right.

image courtesy [55]

This brings us to FastSLAM, developed by Montemerlo and Thrun *[55]*, which only increases in complexity logarithmically, allowing for a far greater number of landmarks to be accounted for in the same amount of computation time. This function utilizes the Rao-Blackwellized particle filter, rather than the Monte Carlo method, to improve robustness to ambiguous data received by the sensors. This method is also sensitive to errors in measurement, which makes it unclear which of several nearby obstacles caused a reading, or errors in motion of the robot, in which it incorrectly estimated its own orientation relative to the objects that it has measured. The latter is almost impossible to compensate for, and it is one of the few downfalls of this method.

One way to reduce error is to employ several robots to

concurrently measure the environment using SLAM or FastSLAM techniques. When the mobile robots measure each other, they are able to reduce the noise in their other measurements. Additionally, data acquired by one robot can be conveyed to others who have yet to explore that region of the environment, giving them an advantage when localizing and updating their own map. This cooperation can effectively increase the efficiency of the mapping process, by allowing a much greater area to be covered in a given amount of time by a group of robots than could be done by a single robot, and with greater accuracy. Mataric has reported on the effectiveness of such collaboration, showing the number of benefits to be reaped by cooperative efforts.

Collaborative use of FastSLAM techniques would be perfect for Robotany's situation, where multiple robots are deployed in the environment, each trying to find an optimal goal state while remembering their origins. This would be especially helpful when it came to the task of seeking water. When the first robot found its location, it would be able to convey that location to all the other robots by means of their common map created through SLAM.

2.6.2 Localization with 802.11b Signal Strength

When groups of multiple autonomous robots collaborate on a SLAM-type task, the means of communication available to them are widely varied. One option that takes advantage of existing infrastructure is that of Wi-Fi, or wireless Ethernet. As discussed by Howard *[37]*, this form of communication can also serve as a means to enable collaborative localization between these robots by sharing data about relative signal strength from various sources as the robot moves about the environment. In one set of tests, known distributions of wireless signal strengths were used to provide initial maps of the environment. The robots were able to navigate based on sensor readings, which were consistent with the *a priori*

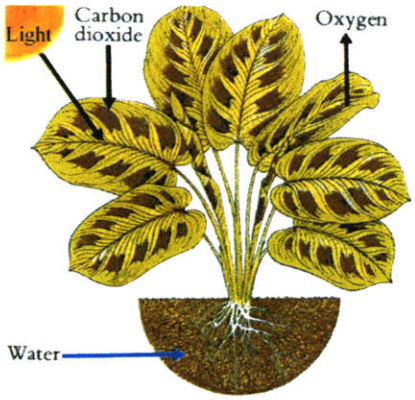


Figure 2.29: Photosynthesis

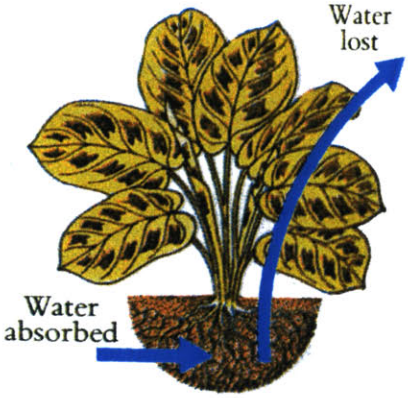


Figure 2.30: Transpiration

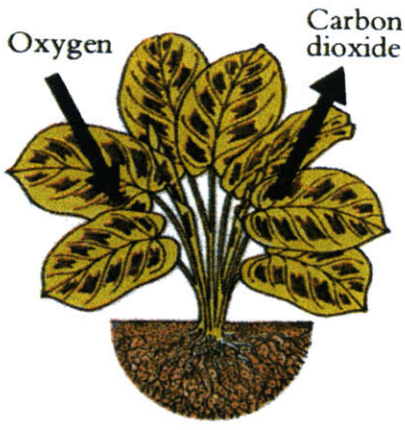


Figure 2.31: Respiration

images courtesy [59]

information. This shows that it is possible to use readings of 802.11b signal strength in conjunction with traditional sensors, such as laser rangefinders, to perform the SLAM task.

2.7 Plant Care

The selection of behaviors evoked by a Robotany vehicle is determined by the needs of the houseplant that rests on top of it. That species' requirements for sunlight and water will affect the homeostasis system that governs the behavior system, which in turn determines Robotany's actions minute to minute. To understand the range of behaviors that will arise, we must first understand the range of needs of the plants for survival. Preferred conditions are generally determined by the native environment of a plant. Although they are highly adaptable to adverse conditions, many plants will not survive in conditions that are too far different from those found in their natural habitat. Presented below is a summary of houseplants' needs adapted from *[59]*.

2.7.1 The Basics: How Plants Work

Plants rely on a process known as photosynthesis to provide energy for all functions that maintain life and promote growth. Photosynthesis occurs when sunlight impinges upon the green leaves of a plant, interacting with the chlorophyll inside each of the cells, and also requires the presence of carbon dioxide and water. The plant takes in carbon dioxide from the air through pores in its leaves, known as stomata, and absorbs water and minerals from the soil through its roots. It then uses these elements to manufacture sugars, which are used in all of its vital processes. Oxygen is released by the plant as a waste product of this process.

The stomata in a plant's leaves remain wide open during active hours in order to absorb as much carbon dioxide from the air as possible. As a side effect, the plant leaves itself open to lose water to the environment through evaporation. If the

root system is unable to find sufficient water to compensate, the plant will begin to wilt as a result. This process of pulling water from the roots to the stomata to the atmosphere is called transpiration. To avoid catastrophic effects, it is recommended to keep the humidity around plants as high as possible. This can be achieved by placing a dish of water under the plant or misting its leaves directly.

Finally, a process known as respiration occurs at all times to aid metabolism of the sugars produced during sunlight hours. Through the same stomata theat absorb carbon dioxide and emit oxygen as a part of photosynthesis, the plant takes in oxygen from the environment and emits carbon dioxide as waste product during respiration. During daylight hours, the effect of photosynthesis overpowers that of respiration and more oxygen is emitted than absorbed. When photosynthesis stops, however, respiration continues.

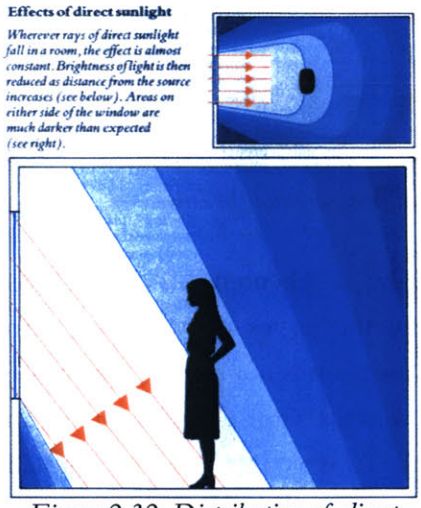


Figure 2.32: Distribution of direct light.

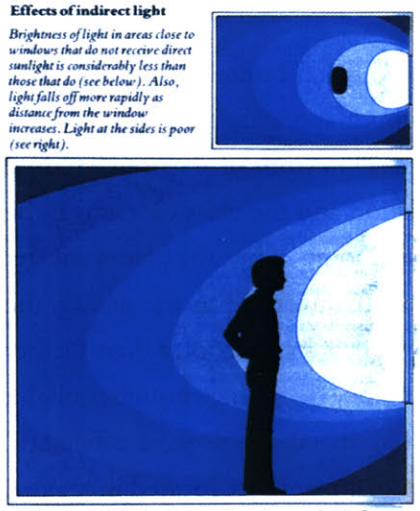


Figure 2.33: Distribution of indirect light.

images courtesy [59]

2.7.2 The Effects of Light

Most plants prefer very specific amounts of sunlight in order to flourish indoors, depending on their origins. Some, such as cacti, prefer direct sunlight and others, such as those that are naturally found on rainforest floors prefer shade. When brought indoors, most plants have difficulty receiving sufficient amounts of sunlight due to the way that intensity of sunlight coming in through a window drops off and changes during the day. Figures 2.32 and 2.33 illustrate the variation in brightness levels of direct and indirect sunlight through a window.

In a large home, it may be possible to find lighting conditions to suit all plants. In an apartment with limited windows, however, this can be more difficult. Additionally, one must take into account the aesthetic influence of where the plants are placed around the home. Having all of one's plants

clustered on a windowsill does not necessarily improve the decorative quality of the room, as many plant owners desire. Figure 2.34 shows the lighting conditions in various places around the home, some of which benefit certain species of plants more than others.

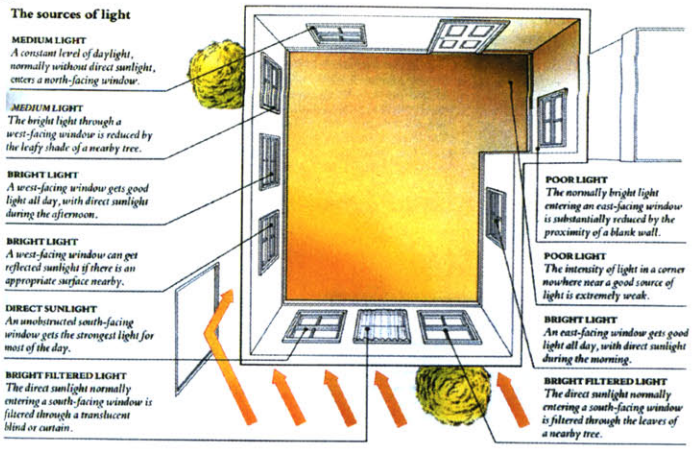


Figure 2.34: Light levels around the home.

As plants seek out sunlight, a phenomenon known as phototropism occurs, where the leaves of the plant turn toward the light source. If the plant is not turned, then all of its leaves and stems will grow in the same direction, giving an unbalanced look to the plant. To avoid this, the plants should be rotated with their pots periodically to ensure even and balanced growth.

Another problem encountered when bringing plants indoors is that when exposed to direct sunlight, they can become scorched. When placed on a sill in front of a closed window, there is not sufficient ventilation and the glass blocks out the ultraviolet end of the light spectrum. Thus the plants will get too hot and can be damaged. Conversely, in the winter, the temperature near a windowpane is much cooler than further into the room. Thus the plant may be getting enough sunlight, but is too cold to carry out its metabolic process efficiently. Great care needs to be taken that plants are located in regions of comfortable temperature, as well as with adequate light levels.

image courtesy [57]

2.7.3 Proper Watering

Perhaps the most crucial aspect of plant care is to give the appropriate amount of water at the appropriate time. While plants are somewhat forgiving if they don't receive the proper amount of sunlight, a plant's health will quickly deteriorate without water. The amount of water is specified by the plant species, ambient conditions, and changes with the changing seasons. For instance, some plants prefer to be kept evenly moist while others prefer to dry out completely before being saturated with hydration. Knowing which type of plant is at hand is most of the battle, but maintaining proper conditions across several different species can be time consuming for a single caretaker. For Robotany, keeping track of an individual plant's needs is embodied in the programming of its support vehicle, making the caretaker's duties lighter while ensuring the survival of its plant to the best of its ability.

2.8 Putting it All Together

The seemingly disparate topics discussed in the previous sections fit together as parts of Robotany's overall makeup when combined in a natural environment. They combine to complement and reinforce each other to build up complex behaviors.

In order to seek out sunlight, Robotany must utilize a light-following routine in conjunction with an obstacle avoidance routine. One or the other alone does not fulfill the goal of bringing sunlight to the plant effectively. Rather, they work in parallel and independantly of one another, like layers in a subsumption architecture (section 2.1.3), to help the robots safely arrive at a satisfactory destination.

Using Braitenberg's approach (section 2.1.2) to seek out sources of light, a vehicle will find the one best solution to any lighting configuration in an empty room. This is independent of the initial conditions in the region when only one light source can be

detected. In the case of multiple light sources, each will have a sphere of influence, within which that source will dominate the effect on the navigation of the vehicle. The algorithm can be shown to be contracting locally for a region of initial conditions. This can be observed in simulations[◊], and shown mathematically.

The obstacle avoidance tactic implemented for Robotany combines the simplicity of the binary cell-occupancy methods, used in potential fields methods (section 2.2.1), with reaction-based AI, such as that used by Genghis (section 2.1.3), to relate distance sensor readings directly to heading changes. This results in a smooth and natural path similar to that described by Fajen et al., but without explicitly using differential equations. By utilizing a simple infrared distance sensor to create a rudimentary map of the obstacles in the environment, a vehicle can steer away from detected obstacles, as would happen in the potential field method. Converse to the abrupt changes in heading specified by the potential fields method however, the parabolic boundary condition for obstacles used by Robotany, combined with the non-holonomic steering abilities of the robot, result in a more gradual turn away from the obstacle. Once the robot has cleared the obstacle, the light-following behavior resumes. The linear relation between light sensor reading and motor speed on each side ensures a smooth transition back to the original heading value.

The homeostatic system is also tied into the light-following and obstacle-avoiding behaviors of Robotany. Homeostasis is integrated into Robotany's programming as a way to arbitrate between light-seeking and water-seeking behaviors. It is able to keep track of the amounts of light a water received by the system, and alter its behavior as the need arises. One way that it alters its behavior is by changing the size of the boundary at which obstacles are detected in the obstacle avoidance routine. For instance, if a dearth of light has been received, the homeostatic routine will shrink the parabolic boundary, causing the robot to

[◊]Simulations of Braitenberg's vehicles abound on the internet. They have been implemented in languages ranging from ANSI-C to Shockwave.

get closer to obstacles and perhaps find new routes through the environment.

From the above discussion, it is easy to see how the individual fields of behavior-based AI, obstacle avoidance, homeostasis and modeling can be related in their differing influences on Robotany's programming. As each varies in rigorousness and similarity to nature, they complement one another to result in a unified stance.

CHAPTER 3

IMPLEMENTATION

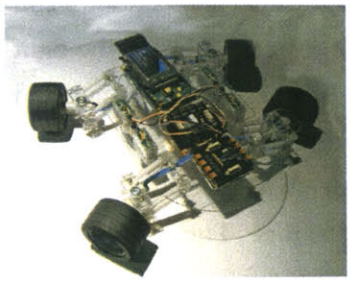


Figure 3.1: One of the robots.

This section covers the design details of Robotany, from the physical form, to the electronics, to the functions that govern its behavior. Analogies are made to natural systems by alluding to these as the body, nervous system and brain, respectively. The current state of implementation, as well as the functions that would need to be implemented to ensure complete autonomy of the robots, are discussed and analyzed.

3.1 The Body

The physical embodiment of Robotany is a system of four-wheel drive, four-wheel steer vehicles. These vehicles can also be used in a wide variety of applications, both autonomous and non, per a researcher's requirements. The strength, ease of manufacture, flexibility of design and low cost are all factors that make Robotany an ideal platform for testing and research. Discussed below are the specific attributes that make this so.

3.1.1 Chassis

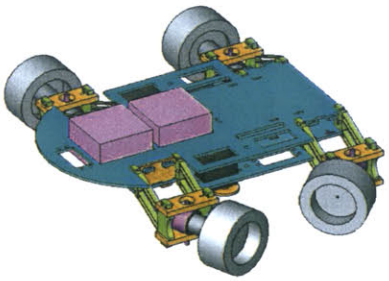


Figure 3.2: A solid model of the robot, as designed using Solidworks.

The chassis of a Robotany vehicle serves to protect its electronics while stabilizing the houseplant that rests on top iof it. It is constructed from sheets of acrylic, also known as Plexiglas. This material was chosen because of its availability, cost, relative strength to thickness ratio, and ease of machining. The entire robot was designed in Solidworks (a solid modeling CAD computer program) before any construction began. By taking advantage of Solidwork's ability to assemble parts, much of the debugging of the physical design can be taken care of in a digital form, eliminating the need for costly physical prototypes. Due to the imprecision inherent in most manufacturing, a little trial and error is an unavoidable step before coming to the final design.

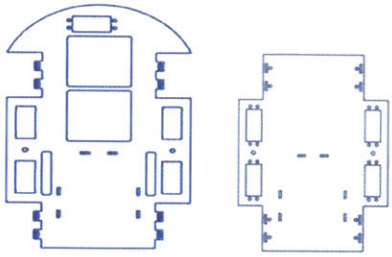


Figure 3.3: The top (L) and bottom (R) plates of Robotany.

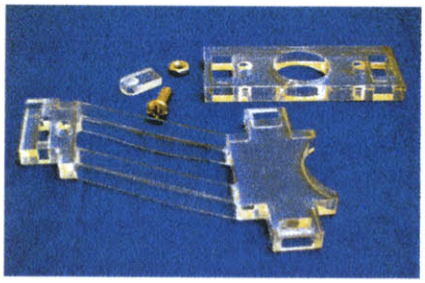


Figure 3.4: Component pieces used in forming a right-angle joint from planar parts.

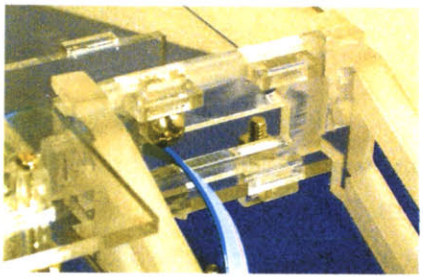


Figure 3.5: Two pieces of a robot's chassis joined at a right-angle using a tab-and-slot method

The parts designed on the computer were cut using a laser cutter. This machine uses a highly focused beam that traces along curves drawn in a computer file to cut material to the desired shapes. The cut edges are very smooth and complicated geometries are simple to realize due to the tight tolerances of the cutting beam. In this way it is quick and easy to get high quality parts that mate together as designed in the ideal world of the computer. See Figure 3.3 for a view of the top and bottom plates of the robots. The ease of using the laser cutter makes it simple to modify a part for a given application. Being able to prototype high quality parts rapidly makes for a swift design process and saves money and time as well.

The only drawback to the laser cutter, which counters its speed and ease-of-use, is that it can only make planar parts. For this reason, the different pieces of Robotany are put together with a tab-and-slot method, reinforced with nuts and bolts. See a detailed picture of the mating in Figure 3.5. An advantage to this method is that it is very secure and robust to imperfection. Since the tolerance on sheets of cast acrylic can vary from batch to batch, it is wise to design interfaces that can accommodate such faults. The tab and slot method does just this, but letting the nuts and bolts specify final alignment of the pieces, rather than the cuts in the acrylic alone. Permitting some inaccuracies and designing for their possible occurrence allows the body of the robot to be constructed under less stringent conditions while producing consistent results.

3.1.2 Suspension

Many vehicles incorporate suspension of some sort into their drive system to minimize the effects of uneven terrain and to cushion the vehicle from shock. As a Robotany vehicle is expected to operate in a natural (and possibly cluttered) environment, it is likely to encounter hazards such as electrical

cords,clothing or other surface irregularities. To mitigate these hazards, Robotany vehicles also use suspensions at each of the four wheels.

Figure 3.6: A LEGO suspension, modeled after traditional piston-and-spring styles.

Whereas most cars, including remote control models, use some form of a spring and a damped piston to provide shock absorption (Figure 3.6), Robotany vehicles use a novel and easy-to-make suspension design. This design was an adaptation of flexures, which are commonly used in the field of MEMS (Micro Electrical Mechanical Systems).*[64]* While MEMS devices are built typically in dimensions on the order of microns, Robotany's suspension has been scaled up to the macro world to provide similar range of motion and damping as model car suspensions at a fraction of the cost. These pieces are made from a polycarbonate material called Lexan, which is very strong, in the materials science sense of the word. This means that the material is elastic, being able to deflect and return to its original position, and durable, being able to endure significant loads while deflecting without fracturing. This material also has good fatigue performance through many cycles of loading. Add to these traits the ease of creating these planar parts on a water jet and low cost of raw materials, and the result is an ideal part for this application.

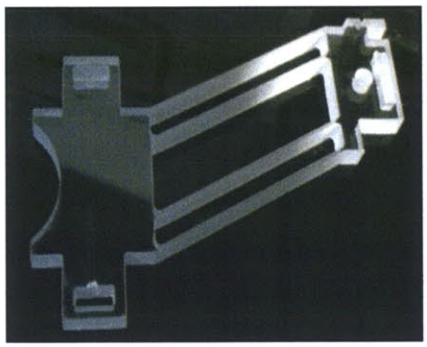


Figure 3.7: Robotany's suspension

The suspension pieces are also planar, cut with a tightly focused, high-pressure beam of water surrounded by fine garnet particles. They are used in pairs at each of the four wheels to increase stability to torsion and stiffness in deflection. See Figures 3.8 and 3.9 for a detailed view of how they are incorporated with the rest of the robot, and how they change from the undeflected to deflected states as needs arise.

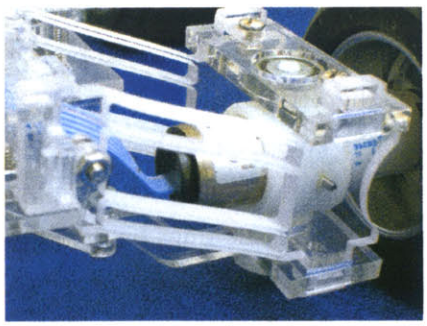


Figure 3.8: The undeflected, rest state of the suspension.

The suspension designed for Robotany is truly unique and efficient. The miniature shocks traditionally used in remote control car models cost on the order of five dollars each.

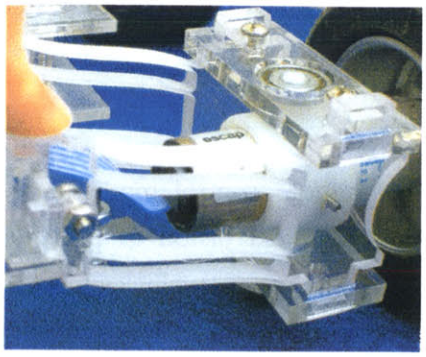


Figure 3.9: The deflected state of the suspension. Note the range of displacement, comparable to traditional suspensions.

Robotany's, on the other hand, cost a fraction of a penny. Given the comparable range and durability of performance of these parts, the cost savings is tremendous. Thus these parts would be ideal to incorporate into other vehicle models as well.

3.1.3 Steering

Each Robotany vehicle contains a four-wheel steering system, meaning that each of its four wheels can be actuated independently. The software controls the angle to which each wheel turns, but the mechanics of the system allow for a range of steering modes.

Each wheel is steered using a high-torque Cirrus BB80 servo motor. This servo drives a four-bar linkage system that connects to the hub of the wheel, as seen in Figure 3.10, causing it to turn. The linkage is designed such that the input angle commanded by the servo motor is the same as the output angle seen at the wheel. The design could be altered to change that relation, allowing for a greater range of motion at the wheel or finer control of the output angle if desired. These linkages are custom made on the laser cutter from acrylic material, and use nuts and bolts at the pivot points.

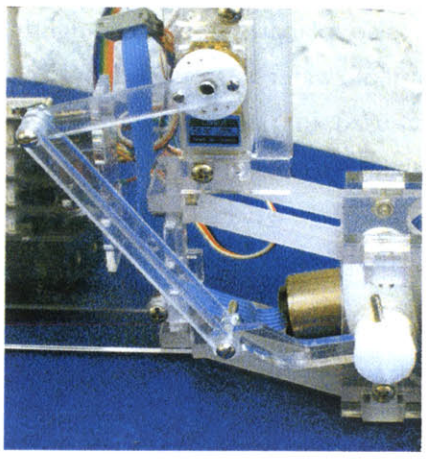


Figure 3.10: Display of the four bar linkage used to turn the wheels during sterring

Drive motors are located at each wheel to provide direct drive functionality, reducing the need for complicated power transmissions and improving the efficiency of power use. Each Portescap 16N28 motor is equipped with a 0.3 Nm gear head and a 16 count encoder to monitor velocity output. The drive motors are mounted in to plastic rings by press fit and a set screw (see Figure 3.11 for detail). These rings are made of ABS plastic and manufactured on a machine called a Fused Deposition Modeler (FDM). This machine takes a file of a solid model, created in Solidworks in this case, and builds it up layer by layer. Each layer is 0.012" thick and made of heated strands of plastic. By fusing these layers together

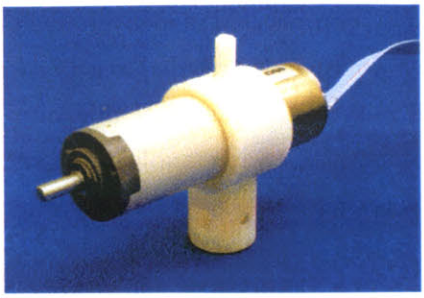


Figure 3.11: A DC motor in its support ring.

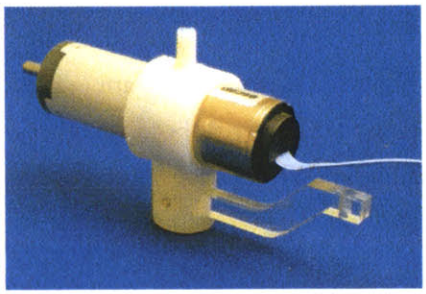


Figure 3.12: The support ring mated with one of the linkages from the four-bar.

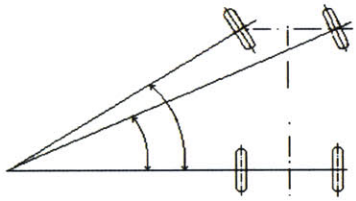


Figure 3.13: Ackerman steering, where the inner and outer sets of wheels travel along circles of different radii.

as they are laid down, very complicated three-dimensional structures can be constructed, that may be impossible to make using conventional tools.

There is a post that comes out of the top of each motor ring and is press fit through a ball bearing mounted in plastic. This bearing reduces the rotational friction and stabilizes the ring about the axis of rotation. The bottom of the ring has another post coming down. The end of the four-bar linkage then passes through this post, causing the motor and therefore the wheel to change orientation (see Figure 3.12). Because of the design of the geometry of the steering system, as the wheels change direction, several different modes of steering are available to the robot.

The first mode of steering, which Robotany uses most often, is the same that automobiles use – Ackerman steering. Ackerman steering takes into account the width of the driving vehicle when determining how much to rotate each wheel. Because each set of wheels – the left side versus the right side – travels along circles of different radii, they need to be turned to different values. See Figure 3.13 for an illustration of this notion. Although Robotany turns all four wheels, Ackerman steering can also be achieved when only two of the four wheels are allowed to turn. This allows for another sub-mode for the case that the robot is made with only two steering motors rather than four, perhaps to save on cost.

Another mode is turning in place. By turning all of the wheels so that their transverse directions all lie on the same circle, the vehicle can negotiate a turn in place. This is a very important achievement when realizing the difference between holonomic and non-holonomic steering geometries. Holonomic steering means that the vehicle is physically able to go in any arbitrary direction independent of current orientation. Most vehicles are not holonomic. For example, most automobiles cannot

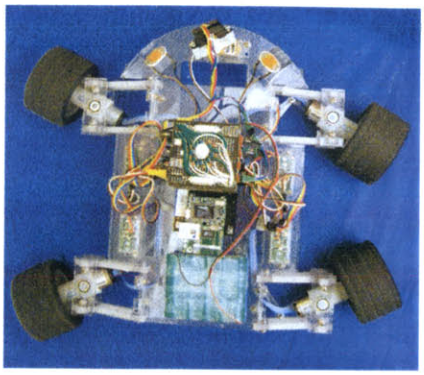


Figure 3.14: One of the robots using Ackerman steering.

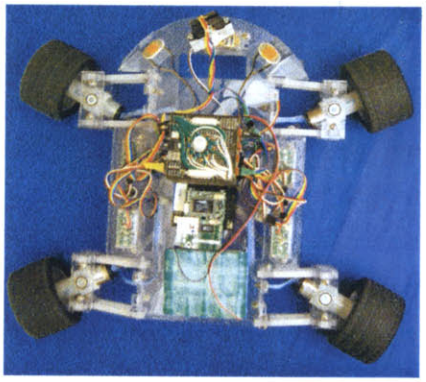


Figure 3.15: The robot demonstrating turning in place.

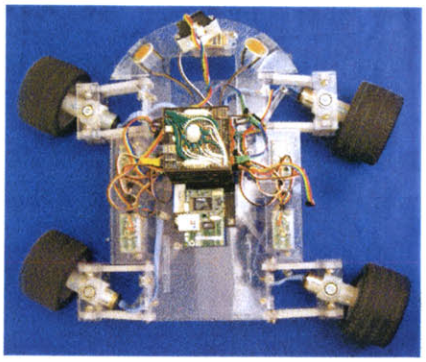


Figure 3.16: One of the robots travelling in a straight diagonal line.

move directly sideways. Instead, complicated maneuvers such as parallel parking and three-point turns are needed to achieve such position changes. The ability to turn in place does not make Robotany truly holonomic. However, it is able to follow similar paths as holonomic robts take, and is free to navigate in much tighter confines as a result.

The third mode of steering is driving in a straight line that is not collinear with the current heading. This is to say that the vehicle can travel along a diagonal line while maintaining heading to the front. See Figure 3.16 for a diagram. This particular mode is not especially helpful to Robotany's goals, but is proof of the great flexibility of the steering system as it was designed. This mode could be used to help the robot fit between two near obstacles if it didn't have room to make a proper Ackerman turn.

The last mode of steering doesn't take advantage of servo motors at all. Instead it utilizes differential steering, the same as used on tanks for maneuvering. Differential steering, also called slip steering, occurs when all wheels are pointed forward, but the wheels on one side rotate more slowly or in the opposite direction from the wheels of the other side. This is how Robotany approaches areas of light while following Braitenberg's conventions. As more light is sensed on the left sensor, the drive motors on the left side spin more slowly. The motors on the right side move faster, and in this manner affect a turn to the left, toward the light. In general, this is not a very accurate mode of steering. Since the technique moves the robot by letting the wheels slip, accurate odometry cannot be achieved. In the absence of feedback control, which is missing from the light-following routine, it is impossible to get an accurate picture of the path taken by the robot. Fortunately, Robotany's navigation methods do not depend on detailed odometry measurements.

3.2 The Nervous System

The nervous system of Robotany is embodied by the electric current passing from the microprocessor brain to the muscles and sensory organs that allow the robot to be situated in its environment. This setup is controlled by the Tower System, described in further detail below. It also consists of the myriad sensors used to gain information about the world around the robot, and the motors, which allow the robot to interact with its surroundings proactively.

3.2.1 Tower System

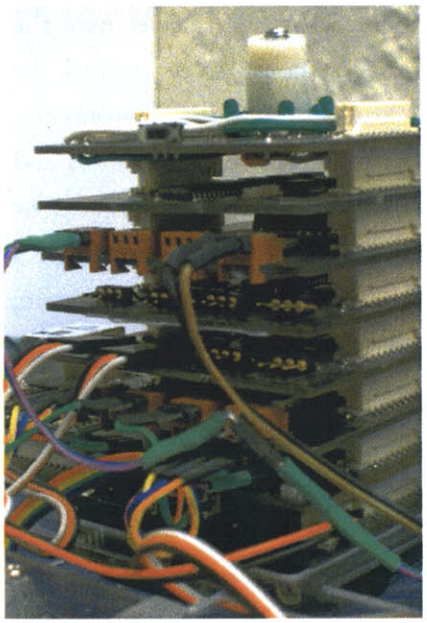


Figure 3.17: The Tower layers used for Robotany.

The Tower System[◊] is a modular electrical development system created by the Grassroots Invention Group (GIG) at the MIT Media Lab for designing and prototyping computational devices. Physically, the Tower consists of a primary foundation layer with a central processor. Robotany uses a foundation equipped with the Rabbit™ 2300 processor from Rabbit Semiconductor™. Functionality is then added by placing stock modules on the stack as needed, and a special prototyping layer allows for simplified design of new modules for the system. Currently, layers are being used that allow for sensor readings (including a custom-built compass layer), servo-motor control, DC motor control, memory storage, and battery charging.

The Tower is programmed in RabbitLogo, also developed by GIG, which is designed to be an easy-to-use programming language that hides the mundane low-level protocols. This abstraction frees the programmer from getting bogged down in the details and allows her to explore higher-level programming concepts sooner, and create a functioning program in fewer iterations. The Rabbit is capable of running up to twenty threads simultaneously, allowing several functions to run independently and concurrently. These functions may each access global variables or arrays, and modify them

[◊]For more information on the Tower System, please see *[46]*

according to their routines. For example, in Robotany, one function constantly sweeps the distance sensors over the path ahead, updating elements in an array. At the same time, another functions reads the values in this array to determine if there is an obstacle ahead and where it is relative to the current heading. Rather than making a full sweep and then sending the entire array to the detection function, having the two operate simultaneously allows for a faster refresh rate and a better chance of noticing the presence of an obstacle in order to react appropriately.

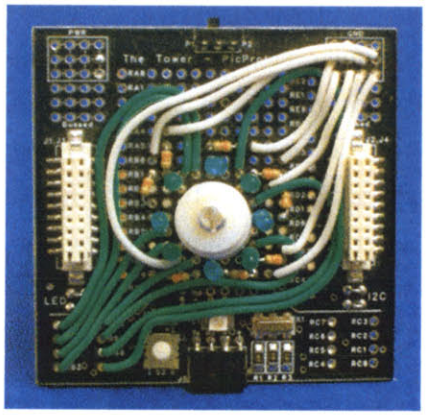


Figure 3.18: The compass layer.

One layer that was custom-developed for Robotany is the Compass Layer (see Figure 3.18). This layer uses the Dinsmore 1655 sensor, an analog compass, to record the heading of the robot as it navigates through the environment. This information is only used for performance analysis and does not affect the robots' behavior. The data supplied by the compass is accessed by the main program through a serial communication protocol used throughout the system, where it is recorded and saved on an EEPROM layer for later retrieval.

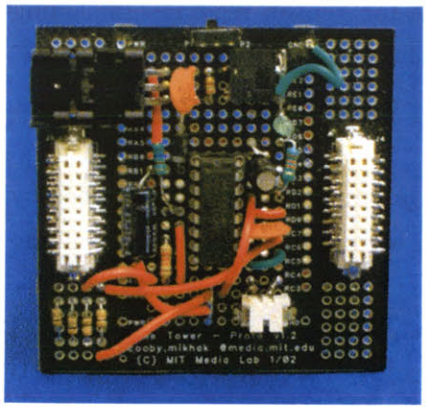


Figure 3.19: The battery charging layer.

Another layer developed for Robotany was the Battery Charging Layer (see Figure 3.19). Here a Maxim™ MAX712 chip uses AC power from a wall wort to fast-charge the custom-built 3000mAh, 6V rechargeable battery packs that provide power for the DC and servo motors. Built from Maxim's specifications on a prototyping layer of the Tower System, this circuit is used off-board from the robot. The eventual plan is to have this layer integrated so that when the robot returns to its base location, where a charging source is located, Robotany will be able to charge itself overnight without assistance.

3.2.2 Sensors

Figure 3.20: One of the photocells used as light sensors on the robots.

Two large photocells are located at the front of the robot, positioned at ±45 degrees from the centerline, and angled upwards by 45 degrees from the horizon. This configuration maximizes the differential in light readings between the two sides. These sensors drive the main behavior of Robotany, finding sunlight in the environment. Using Braitenberg's Vehicle 3a (see Figure 2.8) as the model, the intensity of light sensed on the left sensor inhibits the speed of the left side drive motors. A similar relation exist for the right side. This results in the robot turning toward light sources, and slowing down as the intensity increases. This provides the ideal climate for houseplants as they fulfill their need for sunlight to activate photosynthesis.

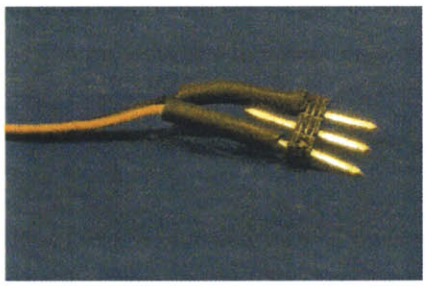


Figure 3.21: A simple humidity sensor.

The humidity sensor is simply two exposed wires submerged in the soil of the plant at a fixed distance. When the soil is moist, the water molecules conduct some of the electricity across the potential between the two wires. When the soil is dry, the potential is much higher and the sensor layer of the Tower registers a different reading. Using this very simple setup, an accurate indicator of the dampness of the soil can be fed into the homeostatic system, which then determines when watering is needed.

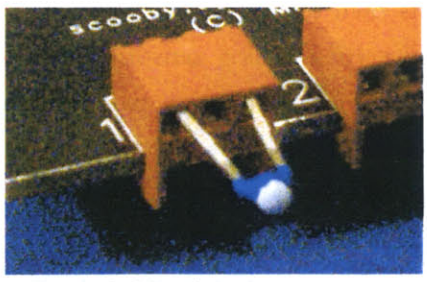


Figure 3.22: A temperature sensor.

A thermocouple is used to sense temperature for Robotany. This sense comes in handy during the summer, when areas of bright light could also be too hot for a houseplant to endure for long periods of time. By monitoring the temperature to which the plant is exposed, the homeostatic system can determine when the plant should seek out cooler locations instead of bright ones.

The Portescap™ DC motors used to drive the robots are equipped with 16 count encoders. Each of these encoders puts out a PWM signal proportional to the speed at which

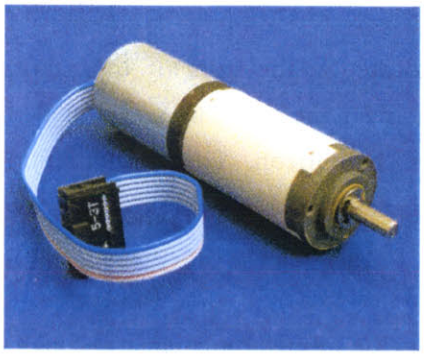


Figure 3.23: One of the Portescap DC motors with gearhead and encoder

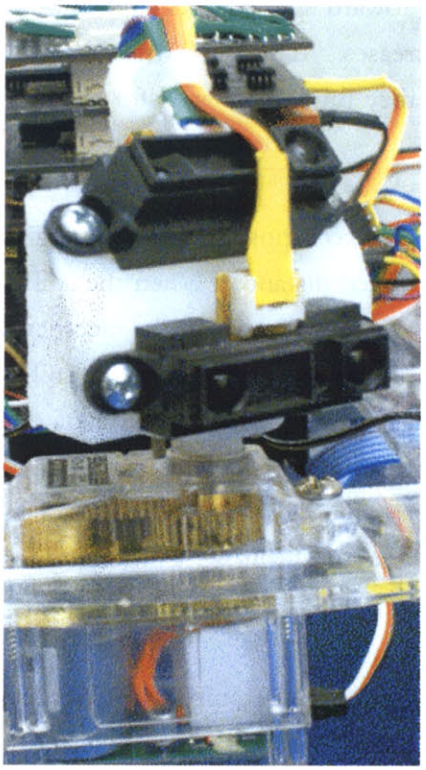


Figure 3.24: The distance sensors mounted to a servo motor.

it is rotating. The signal can be read by a sensor layer and then used in a simple feedback control loop. Controlling the velocity ensures consistent results in the light-following behavior if the terrain changes, for instance from hardwood flooring to carpeting.

Locations of obstacles are sensed with two Sharp™ infrared sensors. Both are located at the front of the robot, mounted onto a servo motor that sweeps them from negative 50 degrees to positive 50 degrees across the bow. One sensor points directly ahead, and is sensitive in the range from ten centimeters to eighty centimeters. This sensor is used to determine if there are any obstacles in the forward path of the robot as it navigates. The other sensor looks upwards by 45 degrees from the horizon and is sensitive in the range of four to thirty centimeters. This sensor is used to determine if the robot is attempting to pass underneath any objects under which the plant would not fit. If this occurs, the robot treats it as an obstacle, reverses course, and navigates accordinglyt.

The path taken by the robot while roaming its environment is captured by the compass layer, in conjunction with velocity readings, and saved on an EEPROM layer on the Tower. These values are then reconstructed using a MATLAB program for reporting purposes. The robot itself does not use this data, as it is not necessary for its own navigation routines. The compass used is a Dinsmore 1655 analog sensor. It uses Hall-effect technology to provide a sin-cosine pair with a voltage swing of 1.3 volts. These voltages can be read by the Tower, recorded, and passed to the MATLAB program, where it is compared to a calibrated curve to return a global heading. Since the time and velocity were also recorded, the complete path con be constructed and used as a measure of performance.

3.3 The Brain

This section covers the problems facing the robots as they try to successfully care for their plant, and the methods used to overcome them.

3.3.1 Seeking Out Sunshine

There are three main components to the light-seeking behavior that Robotany vehicles must embody. They are to detect sunlight, to move toward an area of sufficient brightness, and to stay there so that the plant may absorb a maximal amount for photosynthesis.

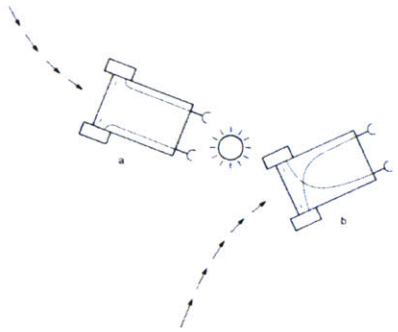


Figure 3.25: Braitenberg's vehicle of type 3a, on which Robotany is modeled.

The simplest solution for finding sunlight is to follow the guidelines set forth by Braitenberg (Section 2.1.2), involving a direct connection between sensor readings and motor output. In this case, Robotany follows the example set by Vehicle Type 3a (Figure 3.25), where the light reading of the left sensor is connected to the left motor in an inhibitory manner, and similarly for the right side. This means that as more light is sensed, the motors will rotate at a slower rate. This affects a turn toward a light source at decreasing velocity with increasing proximity to the source. The raw sensor readings are scaled down so that the highest intensity light causes a reading of zero and complete darkness produces a reading of 255, corresponding to the full range of motor output commands.

Problems arise, however, if areas of brightness matching the maximum cannot be found. If this happens, the robot will continue to roam around and not be satisfied with whatever amount of light is available at the time. For instance, on a cloudy day, it would be impossible to find an area of light bright enough to cause the motors to come to a stop. To resolve this, the robot needs to be able to adapt to current light levels and change the output behavior accordingly. One simple solution is to note if the light levels are falling on both

sensors at the same time. If such a change occurs, then the robot is likely leaving an area of brightness for a darker region. If this is noticed by the program, the robot will stop and stay in the area of brightness. Since this area may be a local maximum, and not the best that the plant could find, search of a new bright spot will be reinstated after a given amount of time. If, however, the decrease in readings is abrupt, then that likely indicates that the robot is merely passing through a shadow and should maintain normal light-seeking behaviors.

3.3.2 Avoiding Clutter

Finding light in an empty room with a single light source is a relatively trivial problem. However, the goal for these robots is to survive in a complex environment, such as a home. This means that the robots will have to deal gracefully with obstacles as they are encountered.

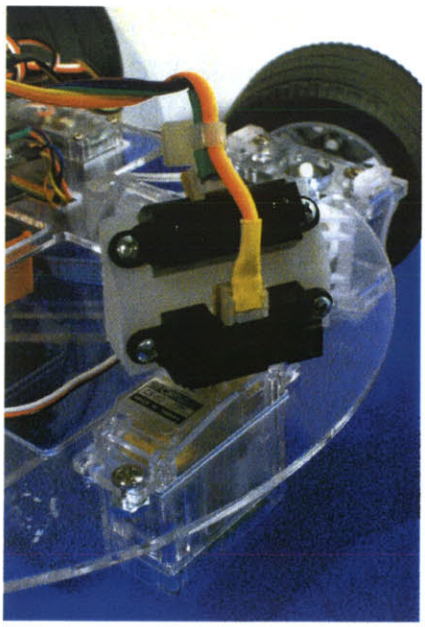


Figure 3.26: The distance sensors, which sweep out the path ahead of the robot.

As Braitenberg's vehicles are not concerned with obstacles in the environment, the routine used for light-following is not sufficient for optimal functionality. To sense the presence of any obstacles in the environment, a single infrared distance sensor is tasked to scan over a given range of angles in front of the robot by means of a servo motor to which it is mounted (see Figure 3.26 for details).

These distance sensor readings are gathered into an array in a separate, constantly running thread on the Rabbit processor. This array provides a rudimentary map of the surroundings, and is accessible to another thread that compares it against a reference array. This reference array corresponds to a parabolic boundary in front of the robot. This geometry effectively ignores an obstacle close to the robot but on its flank, whereas it notices something in front of the robot at a reasonable distance (~40cm). Once the relative location of the obstacle is known, the robot switches behaviors from

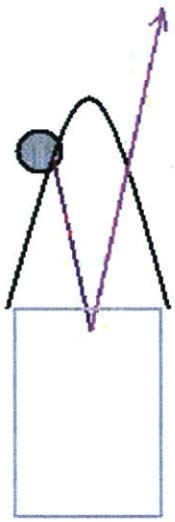


Figure 3.27: A demonstration of how the parabolic boundary is used to produce avoidance motion.

light-following to steering toward a virtual goal. This virtual goal is located in front of the robot at a point diametrically opposite the location of the obstacle. That is, in the robot's reference frame (setting 0° to be directly ahead), an obstacle at $-\theta$° produces a virtual goal at $+\theta$°. Turns are executed at a constant rolling speed using servos to steer the wheels. In this scenario the robot is non-holonomic, meaning it cannot turn in place to immediately produce a change in heading. These traits act in conjunction with the parabolic reference boundary to cause the virtual goal angle to increase as the robot continues in its maneuver to avoid the obstacle. For example, if an obstacle is initially detected at -5°, the robot will begin to head toward a virtual goal at +5°. However, its trajectory must still carry it forward toward the obstacle. Then a flag will be raised, saying that there is an obstacle at -15°, causing the robot to now steer toward +15°. In this manner, the robot's heading will deviate from the original heading in a quadratic fashion and angular acceleration will effectively increase as well.

The obstacle-avoidance portion of the navigation system presented above results in a trajectory which looks very similar to that observed in humans by Warren and Fajen (section 2.2.2). In their model, angular acceleration is high when head-on with the obstacle and drops off to zero when away from it. Although Robotany's function controls heading rather than angular acceleration, the geometry of the parabolic reference boundary produces a similar trajectory. Once the obstacle passes out of the sensing range of the robot, light-following behavior is resumed. When this transition occurs, the robot's trajectory deviates from that predicted by the model developed by Fajen and Warren. This happens because the Braitenberg model used for light-following is first-order, meaning there is no overshoot, while their model is second order. Information about objects behind the navigator is also retained in their model, so there is no discrete change in behavior, like there is

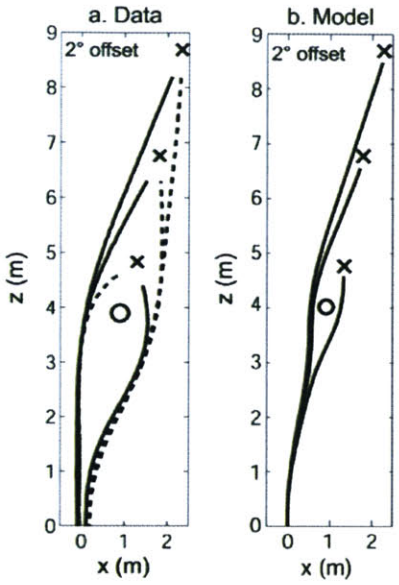


Figure 3.28: The plots obtained by Fajen et al. for humans and that produced by their algorithm.

in Robotany's case.

Differences arise from Warren's approach, which could be attributed to the fact that he uses point mass navigators and obstacles. The dimensions of either object are not factored into his governing dynamics. In Robotany's case, the dimensions do in fact affect the resultant path, in that the robot must compensate for its width and the fact that the obstacle does not exist at one single point. Rather, each obstacle is represented by several units in the obstacle array. Generally, the robot picks the point nearest to 0° to steer away from, rather than which ever edge triggers first, to achieve the quickest avoidance. The comparison routine finds the different chunks of obstacles, treats them as individual objects, and calculates the space between them. This approach differs from that of Warren in that his demonstrated paths only deal with one obstacle and one goal point. The calculation of the results become much more intensive with the addition of a greater number of obstacles. His conclusions state that the algorithm scales linearly with the number of obstacles, and ignores obstacles that are far from the navigator, but the time to calculate solutions increases more rapidly that it does in the routine employed by Robotany.

While this approach to obstacle avoidance certainly seems repeatable, it is not clear if it is also contracting in the fullest sense. It acts like it combines linearly with the light-following behavior; but really, it just turns it off and supercedes control of the robot's trajectory. Warren and Fajen's model, however, is truly a linear combination of both goal and obstacle effects, which is more in line with contraction theory. This may also prove to be its downfall if a robot navigating based on their model ever found itself before a concave obstacle. The force of the goal would continue to pull the robot, while the obstacle (perceived as multiple point obstacles in a concave configuration) repels it. If the concavity of the obstacle were

deep enough, the robot might never turn away from the goal to find its way out, and effectively find itself trapped at the focus of the concavity. Robotany, on the other hand, will allow the robot to find its way clear of the obstacle before seeking out sunlight again. The presence of local minima caused by multiple light sources (e.g. more than one window or partitions created by shadows) also make it unclear whether contraction still holds in this approach, as different initial conditions would lead to different final positions.

In simple environments, the navigation system appears to be contracting in heading. However, in more complex scenes with multiple goals and concave obstacles, it does not. This may be to its advantage however, as a system like Warren and Fajen's could become trapped in an equilibrium point as a result of the same features that cause it to appear to be a contracting system. The simple and organic approach used for Robotany aims to keep things as simple as possible, inherently avoiding explicit mathematical functions that would need to be evaluated at each time step.

In addition to the forward-looking sensor, there is a second sensor looking upwards at 45°. This sensor complements the core obstacle avoidance behavior by detecting if the robot is attempting to head underneath any tables or chairs under which the plant would not fit. If this upward-looking sensor detects any obstacles, the robot doesn't consistently have enough notice to steer around it while continuing in the forward direction. Instead, obstacle detection from above causes the robot to take its standard evasive maneuver of backing up and steering to the left. There is also a simple tilt sensor located on the bottom of the plant's pot, which provides a greater margin of safety. In case the distance sensor doesn't give an indication of an obstacle, if the plant becomes caught on something and begins to tip, the robot will again stop, back up, and continue on to the left. This

redundancy guarantees that the plant will be safe and upright as the robot navigates around the environment.

3.3.3 Balancing Needs

As of yet, the program still needs to track how much light or water that the individual houseplant really needs. To remedy this, a simple homeostatic system has been implemented to track light received and to alter the quality of the behaviors if necessary.

The homeostatic routine monitors the readings on the light sensors over time, and through a simple relation, given below, increments a metric devoted solely to light. In times when the light reading levels are below a threshold value, the routine will decrement this metric. In this manner, if a plant is not receiving sufficient amounts of light, the program will alter the nature of the basic behaviors to help the robot to fulfill its strongest desires. The homeostatic control is given by

$$L = L + p * I * \Delta t \qquad (13)$$

where L is the monitored value, p is a constant, I is the intensity reading, recentered such that a threshold value is equal to zero and anything below that intensity threshold is negative, while values above are positive, and Δt is the time interval over which this measurement takes place. The way that the homeostasis routine affects the behavior of the robot when a strong desire for light is indicated is by shrinking the shape of the parabolic boundary in the obstacle avoidance routine. The default value for the size of the parabola involves a factor of safety equal to two, where a factor of safety of one would correspond to a parabola that extends just past the outer dimension of the robot to the sides, and a distance of 20cm to the front. This is the bare minimum amount of space that the robot needs to be able to maneuver around obstacles. Adding the factor of safety is just that, increasing the amount of cushion that the robot has to navigate through. As the need for light contracts

the parabola, new routes that were previously deemed risky will become available.

Since all plants are not created equal, some basic settings need to be programmed into the robot for each general requirement. For instance, the instructions that come with most plants include vague terms like "partial shade," or "evenly moist." The programming of Robotany is able to convert these terms into threshold values for the homeostatic system. Plants like cacti, which prefer to have their soil dry out completely before being deluged in recreation of their natural desert habitat, would have extremely low thresholds for their water tolerance. Then the homeostatic value associated with wetness will need time to fall to such a low value, at which time it will be ready to accept more water. Also, a plant that needs "full sun" will have a high value relating to sunlight. If it doesn't get enough to maintain its homeostatic ideal, it will change its behavior by shrinking its boundary profile, making new areas that are possibly full of sunlight more accessible.

Sometimes complex behaviors can be derived from simple rules, as advocated by most natural physiologists. For instance, competition for sunlight is a result of two plants with different needs interacting in the same environment. In this example, assume that one robot is acting on behalf of a robust, hearty cactus, while another is acting on behalf of a delicate African violet. The violet will have a very strong desire for bright, filtered sunlight, such as through a curtain. Direct sunlight can burn its leaves, however, so it must maintain a careful balance. The cactus, one the other hand, is relatively ambivalent towards its lighting levels. Although it also prefers bright light, it is much more tolerant of adverse circumstances and decreased lighting than the violet. What happens when these two robots desire the same area of brightness in a given room? Assume that it is the beginning of the day, that the cactus found the bright spot first, and that the violet's

homeostatic levels have caused its parabolic profile to shrink. When the violet seeks out the bright spot, it is willing to get closer to obstacles than the cactus. These robots do not know each other explicitly as separate entities like themselves; rather they are merely other obstacles in the environment. This means that in its quest to attain more sunlight, the violet will approach the cactus. This advance may impinge upon the cactus's obstacle boundary, setting up the obstacle flag. This flag then induces an avoidance maneuver on the cactus's part, thus relinquishing the area of brightness to the violet. In this way, the plant with the stronger desire for the resource at hand pushed the lesser one out of its way. All this without any complicities or agreement between the two.

CHAPTER 4

TESTING AND ANALYSIS

This section covers the successes of the robot and how it achieved its goals. It then discusses features of the robot that have not yet been implemented, but should be to complete the full functionality of autonomously caring for a houseplant.

4.1 Results

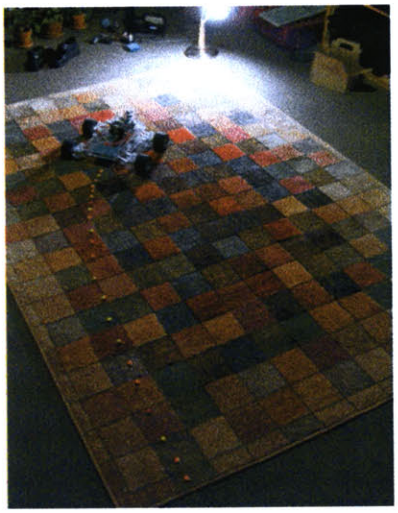


Figure 4.1: The testing environment.

Robotany is able to successfully navigate complicated environments and find its way toward sources of light and stay there, as desired. To show this, several test cases were observed and recorded. Data was originally intended to be recorded by the Tower system and saved on an EEPROM layer. However, compass measurement errors prevented accurate measurement of heading data from being recorded in this manner and forced a return to more primitive methods. Instead, Robotany's scenarios were run on a carpet with a regular grid pattern that permitted a measurement resolution of approximately one inch. Then, as the robot performed its tasks, it was followed by the researcher who dropped markers behind the robot in one-second intervals. Some amount of human error is inherent in this sort of approach, so that the plots for Robotany's progress are not very precise. The uncertainty in this approach to recording data was nevertheless found to be less than the uncertainty in the compass measurements. The general shape of the trajectory and speed presented are qualitatively accurate, but not necessarily quantitatively.

In the first test case, a large area (measuring approximately five feet by seven feet) was illuminated by a single light bulb. The robot was initialized in the opposite corner of the area, facing the opposite wall. The original angle of the light source relative to the robot's initial heading was 30 degrees and the initial distance to the light source was eight feet. Figure 4.1 shows the path taken by the Robotany vehicle. This figure also illustrates the model for

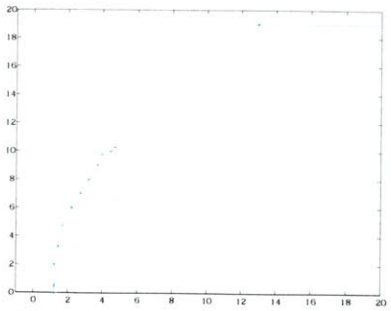


Figure 4.2: The robots path (stars) compared with that generated by Fajen et al.'s algorithm (solid line)

steering presented by Fajen and Warren (as discussed in section 2.3.2).[24,25] As seen from these figures, the paths are similar qualitatively, though they differ in their details. For example, the real robot takes a much more gradual turn toward the light. This difference occurs because the Robotany vehicle uses the non-holonomic method of slip-steering (discussed in section 3.1.3) to navigate toward the light, whereas Fajen et al.'s model is able to respond instantly. The difference is exacerbated by the high rolling friction associated with operating on the carpet. The vehicle's path in Figure 4.2 stops well before the location of the light. This is because the sensors are reading values that are sufficient to cause the robot to come to a halt. Fajen et al.'s model on the other hand, shows the trajectory of their robot to continue through the goal point, as it has no constructs to predict what happens when th goal is finally reached. A similar effect would occur for Robotany if the brightness of the light source were not sufficient to bring the vehicle to a halt. As it is modeled after Braitenberg's Vehicle 3, if the light levels were sensed to be falling, the robot would increase its velocity. To avoid this consequence, Robotany was programmed to sense if the intensity readings were falling on both sensors at the same time. If this happens, the robot determines that it is leaving an area of brightness and stops in its place. This is a slight modification on Braitenberg's model, as the robot now responds to the rate of change of light intensity as well as the intensity itself. After a set period of time (currently only 30 seconds for testing purposes), the robot initiates exploration again in search of a light source in case the one that it was resting in was only a local minima or that lighting conditions had changed.

The first case was run such that the motors could be commanded to operate at the full range of their capabilities. When the light levels are low, the resulting speed of the vehicle is too fast for the distance sensor to update its map and respond to obstacles in a timely manner. To remedy this, the output of the motors was limited to only 40 percent of their ability. This means that although the Tower is capable of sending speed commands as a

number from 0 to 255, corresponding to stopped and full-speed, respectively, Robotany was limited to the output range of 0 to 100 instead. The sensors still sense the same amount of light, but the motors are now prevented from responding, in order to rein in the speed of the robot.

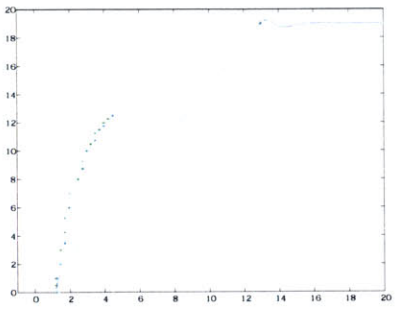


Figure 4.3: Same case as Figure 4.2, but with the robot's wheel velocities limited to 100 out of 255.

The first test case was repeated with the new constraints on motor output in place. The resulting path can be seen in Figure 4.3. Notice that the path taken deviates further from that generated by Fajen et al.'s algorithm than in Figure 4.2. This occurs as a result of the smaller differential between speeds of the left and right wheels, despite a gradient between the sensor readings. The robot travels straight ahead for a longer duration than the previous case because the light readings from each sensor commanded a velocity greater than or equal to 100. As the sensor on the right began to sense more light, it commanded a slower velocity to the right wheels. The left wheels, however, were still being driven at a rate corresponding to a command of 100. This difference still caused the vehicle to turn, but at a slower rate than before. Finally, both light sensors were causing speeds less than 100 and the vehicle completed its turn toward the light.

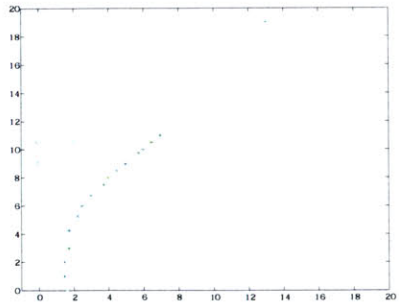


Figure 4.4: Robotany avoiding an obstacle in its original path to the light source.

In the second test, the light and the robot are initialized in the same locations as before. However, this time an obstacle is placed in the path it had taken previously. This obstacle is a toaster (represented by blue stars at each of its four corners in the plots), with a footprint of eleven inches by seven inches, and is placed at an angle and to the left of the path taken in the preceeding setup. As Fajen et al.'s equation of dynamics only deals with point obstacles and point robots, it cannot be compared directly to the results produced by the Robotany vehicle. As the obstacle is detected, the robot activates its obstacle avoidance response and steers away from its perceived location. After it is out of range of the obstacle's influence, it is able to continue along its goal path toward the light. As seen in Figure 4.4, after the robot has turned away from the obstacle, it had overshot the original approach

angle to the ligh source. It can then be seen turning back to the left at the end of its path.

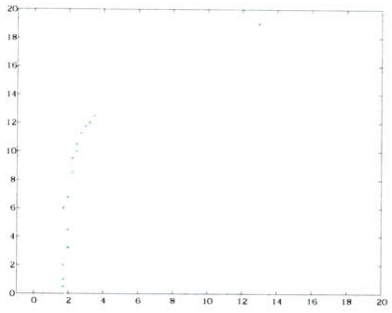


Figure 4.5: Taking an outside route around an obstacle.

The third test case moves the obstacle to the other side of the original path presented in Figure 4.3. In this trial, the robot takes an outside path around the toaster, as seen in Figure 4.5. Instead of using its obstacle avidance routine, however, the robot begins turning away from the obstacle before the distance sensors indicate its presence. This is due to the reaction of the light-following behavior to how the shadow cast by the obstacle affects the amount of light impinging on the photocells. After the robot has passed through the shadow, it begins to turn toward the light again. This time, the distance sensors do pick up on the obstacle, turn to the left a bit, and then turns toward the light, coming to rest as it passes the obstacle on its right.

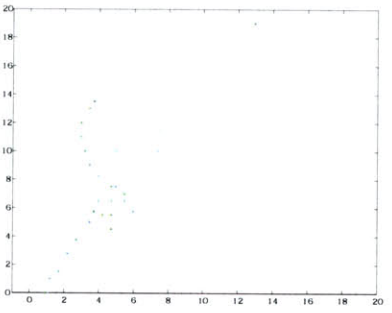


Figure 4.6: Encountering and avoiding an obstacle directly between the robot and its goal.

The fourth test initialized the robot facing the light source, but with the obstacle directly in its path. In this case, the robot was unable to turn away from the obstacle in time to avoid it directly, as shown in Figure 4.6. Instead, the back-up behavior was invoked, twice, before the robot initiated a turn to the left. As the robot passes the obstacle, and out of its shadown, it turns right to approach the light as before. The results of this maneuver could not be reproduced with a dynamic equation like Fajen and Warren's, which has no contingency plan if the robot is faced with an obstacle that it doesn't have time to avoid. In this manner, Robotany is more flexible with respect to the types of obstacles it is prepared to face. This also means that the robot will likely not get stuck in a local minima, such as that caused by a convex obstacle. Figure 4.7 shows a photograph of the trail left by the robot as it navigated toward the light.

Figure 4.7: Resultant path of the robot as it avoids the toaster in its path.

As a final test, the robot was also let free to roam about the unstructured environment of an apartment living room during the daytime. It successfully avoided collisions with furniture and made its way to the windows. Once there, it oscillated between

Figure 4.8: Note the area of darkness below the window in this schematic of light distribution.

the light following and obstacle avoidance behaviors. This is because the floor-to-ceiling windows provided sunlight up to the physical boundary of the wall, which the robot sensed and reacts to. For windows that do not go all the way to the floor, there is an area of muted brightness near the wall, as shown in Figure 4.8. When the robot passes through this area, the drop in light levels would be recognized and the robot would come to a stop.

Telemetry of the robot's path was intended to be obtained by recording compass and velocity data on an EEPROM layer designed for the Tower System, to be used in reconstruction by a MATLAB script. Noise in the communication, possible a result of the motors, caused errors in the recorded compass signal, so the task was moved to a separate Tower foundation.

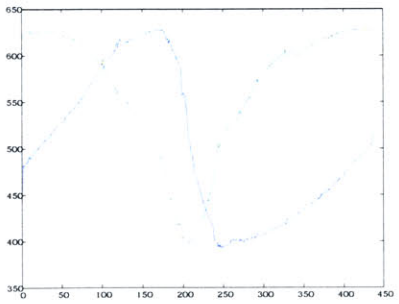


Figure 4.9: Calibration curve created from values put out by the compass as rotated through 360º.

The MATLAB script recreated the path taken by the robot by importing the data recorded by the Towers into arrays. The arrays were compared against a lookup table to determine the angle measured. This lookup table was created by measuring the compass output at each of 360 degrees, as measured by a servo motor. The resulting curves are shown in Figure 4.9. Once the script generated an array of heading values in degrees, these were converted to radians and used with the recorded velocities to plot the path taken in the *x-y* plane. See Figure 4.10 for an example path.

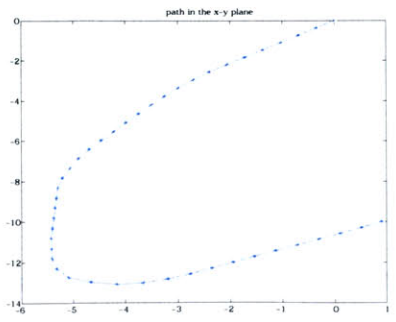


Figure 4.10: Data captured and plotted for a U-turn from southwest to northeast.

Although the data from the compass sensor has a high signal-to-noise ratio, there was a significant amount of hysteresis, which also affected results. For example, when the compass was displaced by 45 degrees counter-clockwise, then returned to its original heading, the output of the channels returned to a lower value than originally recorded. Similarly, when the compass was then rotated by 45 degrees in the clockwise direction and returned, the output was higher than the original value. Fortunately, both channels rise at the same time, causing the entire sin-cosine pair to shift upwards on the y-axis. This meant that the relative reading

remained the same and values could still be obtained by the method described above..

The robots have not yet been tested for several days at a time to investigate the effectiveness of the homeostatic routine on the light-following behavior. Additionally, since water-seeking behavior has not yet been implemented on the robots, the effects of the homeostasis routine would not prove as interesting at this time.

The success of the robots in finding sunlight and avoiding obstacles supports the assumptions made when designing the physical body of the robot. For instance, using a single infrared sensor to scan across the bow of the robot rather than an array of individual sensors proved successful as well as less expensive. More detailed information about the surroundings was obtained as a result as well, since it would not be practical to mount 100 sensors on the robot looking in each of 100 degrees. Also, computational power was saved as the coordination of polling several sensors was not an issue and complicated methods for relating their values to each other were not necessary.

The rechargeable battery pack was a crucial component to the success of the robot. During periods of intensive testing, where motors were running nearly constantly, the 3300mAh battery pack lasted roughly two-and-a-half hours before needing to be recharged. Then, using the fast recharging circuit described in section 3.2.1 restored the batteries to a powered state in a matter of hours.

4.2 Future Work

This section discusses features of the robot that have not yet been implemented, but should be to complete the full functionality of autonomously caring for a houseplant.

4.2.1 Self-Preservation

An important feature that needs to be implemented for Robotany's ultimate success is that of knowing when it is in danger. This danger can come either from external sources, such as a pet or stairs, or internal sources, such as battery drain. Although relatively straightforward in nature, each of these problems can cause complications hindering overall performance.

If a curious puppy or a vindictive cat decides to investigate the robot, how will it handle itself? Although it is very low to the ground and stable on its own right, the pet could easily tip the plant off of its perch, or push the robot off its course. The robot can use the tilt sensor located on the bottom of the plant's pot to tell if it is being disturbed, as it does in normal navigation to detect an attempt at passing under objects. The first recourse can be to emit a sound which is unpleasant to an animal in question. Second would be to back up quickly, and hope that the sudden burst of movement startled the animal into leaving it alone. Of course, a determined animal will not be deterred by these actions, so the robot would then just aim to be still until the animal got bored. Ideally, any owner would first acclimate the pet to the robot's presence before letting it operate without supervision.

As for the presence of stairs in the environment, adding another short-range distance sensor to the front of the robot looking downwards would detect changes in elevation well before the robot was in danger of passing over it. Assuming that the robot is initialized on a flat surface, it can then calibrate its own sensor to determine the default value. Then any deviation can be recognized as an indicator of physical danger and the avoidance routine instated.

A more complicated matter is how to monitor and conserve battery levels. There are chips on the market that provide an

indicator of the voltage output from a battery pack. One in particular is the MAX832 chip, produced by Maxim, which can switch between two separate battery packs when necessary. This chip can also recharge the batteries when is senses the presence of an AC power source. Recognizing when battery levels are low is relatively straightforward using this off-the-shelf microchip. Getting the robot to find an available AC source on its own, however, is a whole other problem.

4.2.2 Returning to Home Base

Having the robot know where it came from and able to return to that same location is a very difficult problem. There are several paths to finding a solution, but determining which path depends on several philosophical factors. For instance, the robot could "remember" where it came from, either by building a map, recording the set of moves that it took to get where it was, or by leaving breadcrumbs. What these solutions have in common is that they all require some form of memory and for meaningful information to be retained for a significant duration. If the decision is made not to rely on complex memory structures, an organic approach could be implemented by installing a beacon of sorts that draws the robot back to its starting point, also the location of its battery charger.

In keeping with the simile to nature that is behavior-based AI, one must first decide if simple creatures use memory or beacons to find their homes after a long day of exploring. Whether animals use landmarks in the environment, remember the movements of their body, or leave signs for themselves along their path varies from species to species. The complexity of these techniques also varies, as does the reliability. For instance, humans are known to navigate by landmarks, but then, we also have very large brains that can remember the details of a wide variety of objects and how to

differentiate them from other, similar objects. Honeybees can communicate the locations of food sources several miles away by way of a complicated "dance" that conveys information about which direction to fly and when to turn. And the humble ant is able to lead others to food and to find its way back to its colony by depositing pheromones along its trail.

Although the method used by the ant to modify its environment in a way only it personally could is the simplest of the three, it is not acceptable for a robot to alter its surroundings in a physical, and therefore sense-able, way. A more practical method might be to install a beacon of sorts at the home location whose attraction supercedes that of sunlight as the batteries run down. Grey Walter took this approach when he was designing his *Machina Specualtrix*. As described in section #, the tortoises were attracted to moderate levels of light. As their batteries ran down however, the photocells became less sensitive and more brilliant sources became attractive. By design, the robots' hutches were lit by a bright bare lightbulb. Thus the hutches became the most attractive area in the environment, and the robots appeared to know to go home for sustenance when their energy levels were depleted. In Robotany's case, an RF beacon could be tuned to each individual robot to guide each back to its home location based on signal strength.

Radio frequency or wireless bandwidth could also be used to transmit data from the robot in the environment to a host computer for storage and map-building. This computer could then transmit directions back to the robot. It would have to know where the robot is through some means of localization, but this has been done via signal strength fields in experiment. If this solution were implemented, it could also transmit information about the weather from the internet to the robot. Then it would know that if it were a cloudy day, not to expect bright sunlight and adjust behaviors accordingly.

This approach is decidedly unnatural, however, and is subject to many unknowns, including how objects in the environment affect the RF or wireless fields. However it is useful to explore technical approaches and then compare them to the way nature has evolved to tackle the same problems now facing mobile robots.

4.2.3 Finding the Elixir of Life

The next most crucial component to the survival of houseplants is finding appropriate amounts of water at appropriate times. There is a slight problem, however, as sources of water in a home are not as ubiquitous as sunlight, and must be detected directly and not merely by proximity.

The first idea for solving this problem, and likely the simplest to enact, is to have the robots carry a reservoir of water around with them. Unfortunately this adds to the weight that the robot must carry around significantly, and the sloshing of water could affect the inertia of the robot as it navigates. If these problems were to be overcome, it would then be a simple matter of actuating a pump when the homeostatic system dictates that would pull water from the reservoir into the plant's pot. The amount of water would be determined by the type of plant and the volume of the pot, both of which would be programmed in when the robot was first used for that plant.

Barring the above solution, another could be to attach a beacon of some sort to a low-profile dish on the ground in the robot's environment. Problems arise though if this beacon-like signal were to be confused with that leading to the base station. If two different signal sources were to be used for the two beacons, then the technology needed would become more complicated as well. The dish would also have to be low enough so as not to trigger the robots' obstacle

avoidance reactions yet hold enough water to be useful to several plants at once. The robots could then dip a hose off the front of the chassis into the water, actuate a pump, and hydrate their symbiont plant. *(Sam) As an alternative to the watering trough approach, one could also rig a water cooler to dispense the liquid through a valve when the robot was sensed to be in the proper position. This idea sounds cool, and avoids having open dishes of water laying about, but the act of orienting the robot properly is itself difficult and pouring water on top of the plant from high above is not an effective delivery method.

Given all of the above considerations, it appears that carrying around a reservoir of water, much as a camel carries in its hump, is the simplest strategy presented thus far. A special fitting would need to be designed to carry the water without possibility of leakage, and higher-torque drive motors would need to specified to carry around the extra load.

4.2.4 Cooperative Navigation

Since it is likely that several robots would be operating in the same physical space, it would be useful if they could communicate information to one another. For instance, if one robot found the water source, it could convey its location to the others, making their search simpler.

In keeping with the behavior-based approach to solving the challenges that face Robotany, one way to communicate this information is to mimic nature's solution used by honeybees. Honeybees use polarization of sunlight through the atmosphere to determine heading and integrate constant velocity measurements to determine distance along their flight path.[#] When they return to the colony, they communicate this information to the other bees by means of a complex dance. The other bees can then use this rudimentary map to

find the same food source. The robots can't dance, but they can potentially communicate similar information by other means.

In order to communicate effectively, the robots need to have a common reference frame. This could be achieved by implementing SLAM on each of the robots. Although this is not an organic solution, the robots would be able to share the information among themselves to create a communally-built map of the environ. This would not only help the group as a whole to find optimal conditions for their plants, but would also increase the likelihood of each returning to its correct starting point. Finding their charging stations is an important aspect of the project, as discussed in section 4.2.2. Several vertebrates are capable of true navigation, of being able to return home after being displaced to a foreign location.[#] A similar ability confined to the space of a house would be of great value to the robots.

Due to limited memory capabilities, instead of retaining complete maps of the environment on each robot, they could also be programmed to send their data to a central computer. This computer could then convey the necessary information to the robots as needed. This centralized approach is inherently in opposition to the natural approach, but may be the only way to overcome the limitations of the current computational platform.

Regardless of the arrangement of information, whether centralized or distributed, the means of communication is still flexible. The robots could be given individual IP addresses and use wireless internet (such as the 802.11b protocol) to talk to each other. The processes for such networks are well established and can be adapted to the robots needs. Another method is for each robot to broadcast their information in radio frequency (RF). Other robots then listen for relevant

information that they can use in their own quest. Although this active listening requires significant resources, it may be worthwhile to save the robot the effort of discovering the entire environment for itself.

CHAPTER 5

CONCLUSIONS

Robotany was designed to enable houseplants to seek out ideal conditions for their survival, within the confines of a household. This system gives the plants the freedom of mobility and provides them with a dedicated caretaker to oversee their needs. By doing so, the system frees the plants' owners from worry and responsibility and encourages the owners to share their environment with other living things.

This task was achieved by incorporating a wide range of artificial intelligence and controls techniques. These disparate topics combined to reinforce one another and to make the program behind Robotany more resilient. From Braitenberg's vehicles and Breazeal's Kismet, to contraction theory and SLAM techniques, all have had their influence on the development of Robotany's character. While the implementation of each aspect varied, they all inspired the development of Robotany's programming.

The robots have been shown to successfully find light and navigate their environment. It is especially useful that they can deal with an arbitrary, unstructured environment without becoming stuck or confused. This flexibility is one of the strengths associated with a behavior-based approach to artificial intelligence. Also, the fact that such complex behavior was programmed on and produced by a relatively low-power computational foundation highlights the simplicity and elegance off most behavior-based approaches.

It is recommended that the programming expand upon Robotany's current abilities to include explicit interactions between the robots and mapping and localization of the environment. Adding these features will enhance the performance and utility of the robots, and make them more welcome additions to the home.

From behavior-based AI to homeostasis to navigation techniques, nature's solutions were the model behind the desired approaches. This is in deference to the millennia that nature has used to evolve beautifully simple solutions to the problems faced by any living creature.

REFERENCES

[1] Agre, P., Chapman, D.; *Pengi: A theory of activity*, Proceedings of AAAI 47, 1987

[2] Arkin, R.C.; *Homeostatic Control for Mobile Robots*, Journal of Robotic Systems, 1992

[3] ASIMO website, *http://world.honda.com/ASIMO/* accessed Jan, 2004

[4] Balch, T., Arkin, R.C.; *Communication in Reactive Multi-Agent Systems*, Autonomous Robots 1, 1994

[5] Borenstein, J., Koren, Y.; *Real-Time Obstacle Avoidance for Fast Robots*, IEEE Transactions on Systems, Man, and Cybernetics, Vol.19, No.5, 1989

[6] Braitenberg, V.; *Vehicles, Experiments in Synthetic Psychology*, MIT Press, 1984

[7] Breazeal, C.; *Early Experiments Using Motivaitons to Regulate Human-Robot Interaction*, Proceedings of AAAI Fall Symposium, 1998

[8] Breazeal, C., Scasselati, B.; *How to Build Robots that Make Friends and Influence People*, Proceedings of International Conference on Intelligent Robots and Systems, 1999

[9] Brooks, R.A.; *Intelligence Without Representation*, Artificial Intelligence 47, 1991

[10] Brooks, R.A.; *New Approaches to Robotics*, Science, p.1227-1232, 1991

[11] Brooks, R.A.; *Artificial Life and Real Robots*, Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, Varela and Bourgine, eds., MIT Press, 1992

[12] Brooks, R.A., Breazeal, C.; *Embodied Intelligence*, MIT Press (forthcoming), 1998

[13] Brooks, R.A.; *Cambrian Intelligence: The Early History of the New AI*, MIT Press, 1999

[14] Camus, T.; *Real-Time Quantized Optical Flow,* The Journal of Real-Time Imaging: Special Issue on Real-Time Motion Analysis, 1997

[15] Cannon, W.B.; "The Body Physiologic and the Body Politic", *Science*, New Series, Vol.93, Issue 2401, p.1-10, 1941

[16] Cavusoglu, M.C., Tendick, F.; Potential Fields Assignment for EE125 at University of California, Berkeley, Spring 2001

[17] Chandler, R.C., Meiszer, K.A., Arroyo, A.A.; *LawnShark: A New Platform for Autonomous Mowing and Navigation*, Florida Conference on Recent Advances in Robotics, 1999

[18] Chang, D.E., Marsden, J.E.; *Gyroscopic Forces and Collision Avoidance with Convex Obstacles*, Proceedings of Conference in Honor of A.J. Krener's 60th Birthday, 2002

[19] Coradeschi, S., Karlsson, L.; *A Behavior-Based Approach to Reactivity and Coordination: A Preliminary Report*, "Intelligent Agents," Vol.IV, Springer Verlag Lecture Notes in Artificial Intelligence, 1998

[20] Dissanayake, M.W.M.G., Newman, P.M., Clark, S., Durrant-Whyte, H.F., Csorba, M.; *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem*, IEEE Transactions on Robotics and Automation, Vol.17, No.3, 2001

[21] Dudek, G., Jenkin, M.; *Computational Principles of Mobile Robots,* Cambridge University Press, 2000

[22] Engleson, S.P.; *Learning Robust Plans for Mobile Robots from a Single Trial*, Proceedings of AAAI/ IAAI, Vol.1, 1996

[23] Esch, H.E., Zhang, S.W., Srinivasan, M.V., Tautz, J.; "Honeybee dances communicate distances measured by optic flow," *Nature*, Vol.411, 2001

[24] Fajen, B.R., Warren, W.H.; *Behavioral Dynamics of Steering, Obstacle Avoidance, and Route Selection,* Journal of Experimental Psychology: Human Perception and Performance, 2003

[25] Fajen, B.R., Warren, W.H., Temizer, S., Kaebling, L.P.; *A Dynamical Model of Visually Guided Steering, Obstacle Avoidance, and Route Selection*, International Journal of Computer Vision, Vol.54, 2003

[26] Falcone, E., Gockley, R., Porter, E., Nourbakhsh, I.; *The Personal Rover Project: The comprehensive design of a domestic personal robot*, Robotics and Autonomous Systems 42, 2003

[27] Fenwick, J.W., Newman, P.M., Leonard, J.J.; *Cooperative Concurrent Mapping and Localization*, Presented at IEEE International Conference on Robotics and Automation, 2002

[28] Ferrari, M., Ferrari, G., Hempel, R.; *Building Robots with LEGO Mindstorms*, Syngress Publishing Inc, 2002

[29] Franke, D., Frick, K., Holm, H., Moor, T.; *Hybrid Resource Allocation Problems - A Laboratory Case Study*, Proceedings of the International Symposium on Advanced Manufacturing Processes, Systems, and Technologies, 1999

[30] Gage, D.W.; *Many-Robot MCM Search Systems,* Proceedings of Autonomous Vehicles in Mine Countermeasures Symposium, 1995

[31] Gat, E.; *Towards Principled Experimental Study of Autonomous Mobile Robots*, Autonomous Robots, Vol.2, 1995

[32] Haag, A., Slotine, J.J.E.; *Contracting Force Fields in Local Obstacle Avoidance*, MIT Nonlinear Systems Laboratory Report, MIT-NSL000601, June 2003

[33] Haddad, H., Khatib, M., Lacroix, S., Chatila, R.; *Reactive Navigation in Outdoor Environments Using Potential Fields*, Proceedings of IEEE International Conference on Robotics and Automation, 1998

[34] Holland, O., McFarland, D.; *Artificial Ethology*, Oxford University Press, 2001

[35] Homeostasis website, *http://www.esb.utexas.edu/palmer/bio303/group32/page3.htm,* Jan, 2004

[36] Horswill, I.; *Specialization of Perceptual Processes,* Ph.D. Thesis, Massachusetts Institute of Technology, 1994

[37] Howard, A., Siddiqi, S., Sukhatme, G.S.; *An Experimental Study of Localization Using Wireless Internet*, Proceedings of the 4th International Conference of Field and Service Robotics, 2003

[38] Howard, A., Matarić, M.J., Sukhatme, G.S.; *Putting the "I" in Team: An Ego-Centric Approach to Cooperative Localization*, Proceedings of IEEE International Conference on Robotics and Automation, 2003

[39] Jones, J.L., Flynn, A.M., Seiger, B.A.; *Mobile Robots: Inspiration to Implementation*, A.K. Peters, 1999

[40] Khatib, O.; *Real-time Obstacle Avoidance for Manipulators and Mobile Robots*, The International Journal of Robotics Research, Vol.5, No.1, 1986

[41] Kleeman, L.; *Optimal Estimation of Position and Heading for Mobile Robots Using Ultrasonice Beacons and Dead-reckoning*, Proceedings of IEEE International Conference on Robotics and Automaiton, 1992

[42] Koshland, D.E.; "The Seven Pillars of Life," *Science*, Vol.295, p.2215-2216, 2002

[43] Kube, C.R., Bonabeau, E.; *Cooperative Transport by Ants and Robots,* Robotics and Autonomous Systems 30, 2000

[44] Latombe, J.C.; *Robot Motion Planning*, Kluwer Academic Publishers, 1991

[45] Little Shop of Horrors, *http://www.imdb.com/title/tt0091419/*

[46] Liu, Y., Thrun, S.; *Gaussian Multi-Robot SLAM*, Proceedings of Neural Information Processing System Conference, 2003

[47] Lohmiller, W., Slotine, J.J.E.; *On Contraction Analysis of Non-linear Systems,* Automatica, Vol.34, No.6, 1998

[48] Lozano-Perez, T., Mason, M., Taylor, R.H.; *Automatic Synthesis of Fine-Motion Strategies for Mobile Robots*, International Journal of Robotics Research, Vol.3, No.1, 1984

[49] Lyon, C.; *Encouraging Innovation by Engineering the Learning Curve*, Master of Engineering Thesis, Massachusetts Institute of Technology, 2003

[50] Matarić, M.J.; *Issues and Approaches in Design of Collective Autonomous Agents*, 1994

[51] Matarić, M.J.; *Interaction and Intelligent Behavior*, Ph.D. Thesis, Massachusetts Institute of Technology, 1994

[52] Matarić, M.J.; *Situated Robots*, Encyclopedia of Cognitive Science, Nature Publishers Group, MacMillan Reference, Ltd., 2002

[53] Matarić, M.J. website, *http://robotics.usc.edu*, accessed Jan, 2004

[54] Menzel, P., D'Aluisio, F.; *RoboSapiens: Evolution of a New Species*, MIT Press, 2000

[55] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.; *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem*, Proceedings of AAAI, 2002

[56] Mussa-Ivaldi, F.A., Bizzi, E.; *Motor Learning Through the Combination of Primitives*, The Royal Society, 2000

[57] Parker, L.; *Alliance: An architecture for fault tolerant multi-robot cooperation*, IEEE Transactions on Robotics and Automation, Vol.14, No.2, 1998

[58] Pawson, R.; *The Robot Book*, Windward, 1985

[59] Reader's Digest, *Success with Houseplants*, The Reader's Digest Assn., Inc., 1979

[60] Robotics FAQ webisite, *http://www.fri.cmu.edu/robotics-faq/*, accessed Jan, 2004

[61] Rosenschein, S.J., Kaelbling, L.P.; *The synthesis of digital Bratman: 29 machines with provable epistemic properties*, Proceedings of Conference on the Theoretical Aspects of Reasoning about Knowledge, 1986

[62] Rosenblatt, J., Williams, S., Durrant-Whyte, H.F.; *Behavior-Based Control for Autonomous Underwater Exploration*, Proceedings of IEEE International Conference on Robotics and Automation, 2000

[63] Sibley, G.T., Rahimi, M., Sukhatme, G.S.; Robomote: *A Tiny Mobile Robot Platform for Large-Scale Ad-Hoc Sensor Networks*, Proceedings of IEEE International Conference on Robotics and Automation, 2002

[64] Slocum, A.H.; *Precision Machine Design*, ch.8.6, Society of Manufacturing Engineers, 1992

[65] Slotine, J.J.E., Li, W.P.; *Applied Nonlinear Control*, Prentice Hall, 1991

[66] Telegarden website, *http://www.usc.edu/dept/garden/* accessed Jan, 2004

[67] Tews, A., Wyeth, G.; *Thinking as One: Coordination of Multiple Robots by Shared Representation*, Proceedings of IEEE International Conference on Intelligent Robots and Systems, 2000

[68] Thrun, S., Burgard, W., Fox, D.; *A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots*, Machine Learning and Autonomous Robots, 1998

[69] Thrun, S., Schulte, J., Rosenberg, C.; *Interactions with Mobile Robots in Public Places*, Proceedings of IEEE Intelligent Systems, 2000

[70] Todd, P.M., Wilson, S.W., Somayaji, A.B., Yanco, H.A.; *The blind breeding the blind: Adaptive behavior without looking*, Proceedings of the 3rd Annual Conference on Simulation of Adaptive Behavior, 1994

[71] Tower project website, *http://gig.media.mit.edu/projects/tower/*, accessed Jan, 2004

[72] Valvanis, K.P., Hebert, T., Kolluru, R., Tsourveloudis, N.; *Mobile Robot Navigation in 2-D Dynamic Environments Using an Electrostatic Potential Field*, IEEE Transactions of Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol.30, No.2, 2000

[73] Vaughan, R.T., Stoy, K., Sukhatme, G.S., Matarić, M.J.; *LOST: Localization-Space Trails for Robot Teams*, IEEE Transactions on Robots and Automation: Special Issue on Multi-Robot Systems, 2002

[74] Walter, W.G.; *The Living Brain*, WW Norton and Co., 1963

[75] Walter, W.G. website, *http://www.plazaearth.com/usr/gasperi/walter.htm* accessed Jan, 2004

[76] Weber, K., Venkatesh, S., Srinivasan, M.V.; *Insect Inspired Behaviours for the Autonomous Control of Mobile Robots*, Proceedings of International Conference on Pattern Recognition, 1997

[77] Wehner, R.; *Matched Filters - Neural Models of the External World*, Journal of Comparative Physiology A, Vol.161, Springer Verlag, 1987

[78] Williams, S.; *Efficient Solutions to Autonomous Mapping and Navigation*, Ph.D. Thesis, University of Sydney, 2001

[79] Yamauichi, B., Schultz, A., Adams, W.; *Mobile Robot Exploration and Map-Building with Continuous Localization*, Proceedings of IEEE International Conference on Robotics and Automation, 1998

[80] Yoon, D.Y., Oh, S.R., Park, G.T., You, B.J.; *A Behavior-Based Approach to Reactive Navigation for Autonomous Robot*, Proceedings of the 15th Triennial World Congress, Barcelona, Spain, 2002