Lorigo, L. M., Brooks, R. A. & Grimson, W. E. L. (1997). Visually-guided obstacle avoidance in unstructured environments. *IEEE Conference on Intelligent Robots and Systems* September 1997. Also available at `http://www.ai.mit.edu/people/liana/liana.html`

Maimone, M. (1997). *Lunar Rover Navigation 1996* (Web Page). `http://www.cs.cmu.edu/afs/cs/project/lri/www/nav96.shtml`

Marr, D. & Nishihara, H. K. (1978). Visual Information Processing: Artificial Intelligence and the Sensorium of Sight. *Technology Review, 81,* 1978, pp. 2-23.

Matthies, L. H. (1992). Stereo vision for planetary rovers: stochastic modeling to near real-time implementation. *International Journal of Computer Vision*, 8(1):71-91, July 1992.

Matthies, L. H., Kelly, A., & Litwin, T. (1995). *Obstacle Detection for Unmanned Ground Vehicles: A Progress Report.* Also available from `http://robotics.jpl.nasa.gov/people/lhm/homepage.html`

Meyer, D. E. & Schvaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology, 90,* pp. 227-234

Meyer, D. E., Schvaneveldt, R. W. & Ruddy, M. G. (1975). Loci of contextual effects on visual word-recognition. In P. M. A. Rabbitt, & S. Dornic (Eds.), *Attention and Performance* (Vol. 5, pp. 98-115). London: Academic Press.

Nourbakhsh, I. R. (1996). A Sighted Robot: Can we ever build a robot that really doesn't hit (or fall into) obstacles? *The Robotics Practitioner*, Spring 1996, pp. 11-14.

Pollack, I., & Pickett, J. M. (1964). Intelligibility of excerpts from fluent speech: Auditory vs. structural context. *Journal of Verbal Learning and Verbal Behavior, 3,* pp. 79-84.

Pomerleau, D. (1993). *Neural Network Perception for Mobile Robot Guidance.* Kluwer Academic Pub.

Rumelhart, D. E., McClelland, J. L. & The PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Volume 1: Foundations. Cambridge, Massachusetts: The MIT Press.

Selfridge, O. G. (1955). Pattern recognition in modern computers. *Proceedings of the Western Joint Computer Conference.*

Warren, R. M., & Warren, R. P. (1970). Auditory illusions and confusions. *Scientific American* 223, pp. 30-36.

Yam, P. (1998). Roaches at the Wheel. *Scientific American* 278 (1) January 1998, p. 45

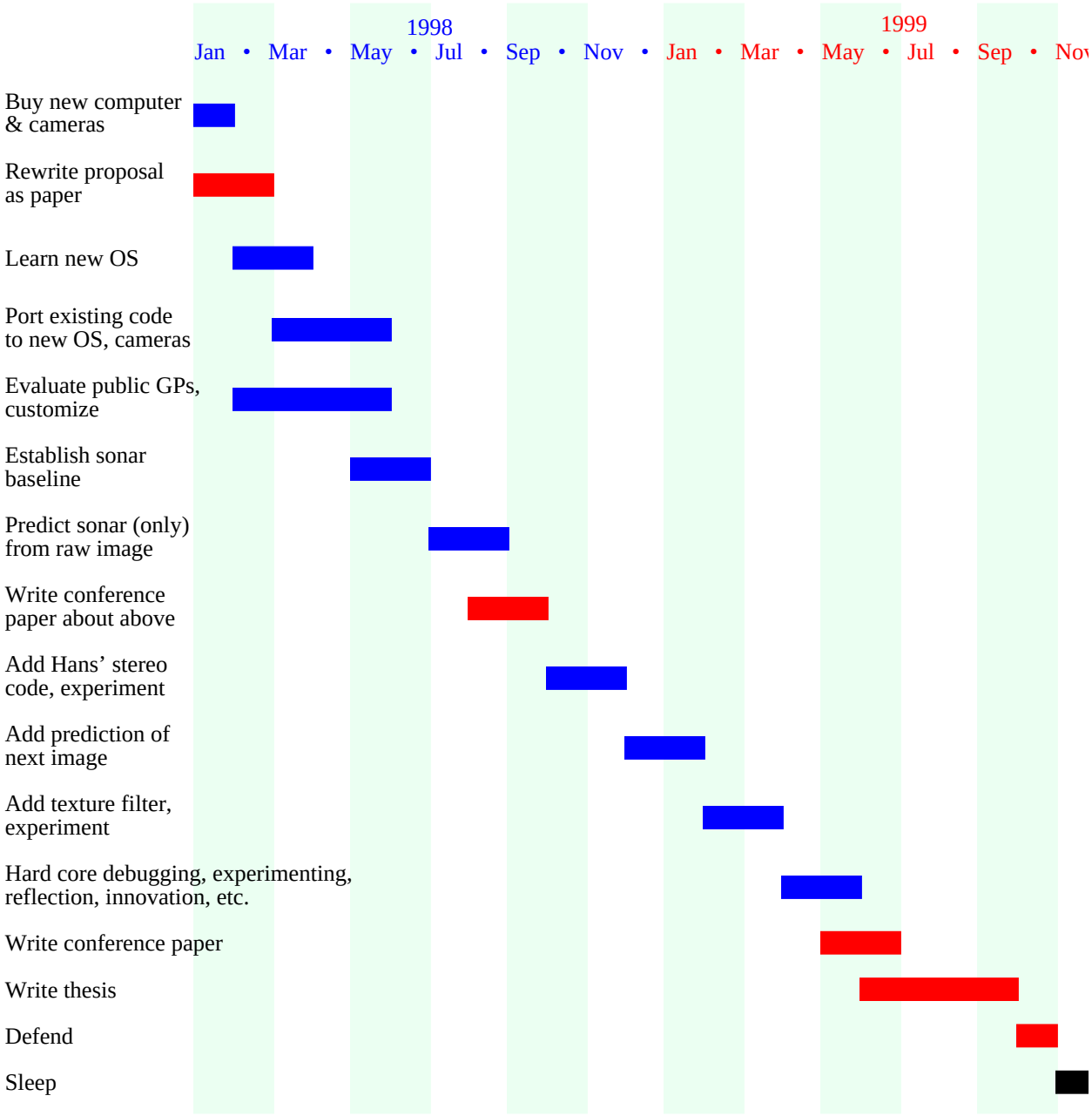


Figure 12: Thesis Timeline

Krotkov, E. & Bajcsy, R. (1993). Active Vision for Reliable Ranging: Cooperating Focus, Stereo, and Vergence. *International Journal of Computer Vision*, 11:2, pp. 187-203

Krotkov, E., Hebert, M. & Simmons, R. (1995). Stereo perception and dead reckoning for a prototype lunar rover. *Autonomous Robots* 2(4) December 1995, pp. 313-331

Lindsay, P. H., & Norman, C. A. (1972). *Human information processing: An introduction to psychology.* New York: Academic Press.

Lorigo, L. M. (1996). *Visually-guided obstacle avoidance in unstructured environments.* MIT AI Laboratory Masters Thesis. February 1996. Also available at `http://www.ai.mit.edu/people/liana/liana.html`

Summary
Genetic Programming, set up to do function regression with an ordered pair twist, will learn programs for both of the triangles in Figure 9. The inputs and outputs are shown in that figure. The entire system of Figure 9 is then evaluated, as explained in Figure 8 and Table 1 on page 20.

8. Deliverables and Timeline

My thesis will have two main deliverables, the first being a refinement of the arguments in Sections 3, 4, 5, and 6, and the second being the implementation and evaluation of the system in the previous Section.

Specifically, I will deliver:

- As part of my written thesis, an expanded discussion of the issues in human and robot perception, in defense of the idea that black box architectures place strong limitations on perception; that this is done in order to manage complexity when systems are designed by hand; and that learning techniques can create systems that can alleviate these problems.
- To attempt to formalize the distinction between hard and soft separation, and any other amenable distinctions.
- To write a program that learns to avoid obstacles for the Uranus mobile robot. The learned program will use vision and dead reckoning as its sole inputs, although the learning algorithm will make use of sonar and bump sensors for feedback.
- To evaluate the above program, recommend possible improvements, and revise my arguments in light of this experience.

The timeline is shown in Figure 12.

References

Ashcraft, M. H. (1989). *Human Memory and Cognition*. Scott, Foresman and Company.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence* 47 (1991) 139-159

Coombs, D., Herman, M. Hong, T. & Nashman, M. (1995). Real-time Obstacle Avoidance using Central Flow Divergence and Peripheral Flow. *Fifth International Conference on Computer Vision* June 1995, pp. 276-83.

Horswill, I. (1994). Specialization of Perceptual Processes. Ph.D. Thesis, Massachusetts Institute of Technology, May 1993. Also available from `http://www.cs.nwu.edu/~ian`

Horswill, I. (1993). Polly: A Vision-Based Artificial Agent. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93),* July 11-15, 1993, Washington DC, MIT Press. Also available from `http://www.cs.nwu.edu/~ian`

Isenberg, D., Walker, E. C. T., Ryder, J. M., & Schweikert, J. (1980). *A top-down effect on the identification of function words.* Paper presented at the Acoustical Society of America, Los Angeles, November 1980. Quoted in Rumelhart *et al.* 1986, *Parallel Distributed Processing* Vol. 1

Jochem, T. (1997). *The ALVINN Project Home Page.* `http://www.cs.cmu.edu/afs/cs/project/alv/member/www/projects/ALVINN.html`

Kanizsa, G. (1976). Subjective contours. *Scientific American* 234:48-52.

Katragadda, L., Murphy, J. Apostolopoulos, D., Bapna, D. & Whittaker, W. (1996). Technology Demonstrations for a Lunar Rover Expedition. *1996 SAE International Conference on Environmental Systems,* Monterey, CA.

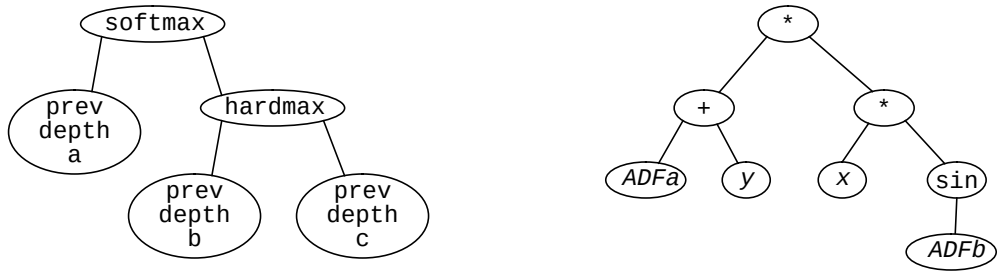
Kinnear, K. E. Jr. (1994). A Perspective on the Work in this Book. In Kinnear, K. E. Jr. ed., *Advances in Genetic Programming,* MIT Press.

Koza, J. R. (1992). *Genetic Programming: On The Programming Of Computers By Means Of Natural Selection.* Cambridge: MIT Press.

Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge: MIT Press.

Table 2: Functions to be used with Genetic Programming

ADF Function	Description
+1, +2, *1, *2, etc.	Versions of all the functions above that work on either the first or second number in the pair, leaving the other alone.
hardmax	Hard Maximum (binary). Of the two input pairs, selects the one with the greatest second value.
softmax	Soft Maximum (binary). Compute $(\|y_1\|(x_1, y_1) + \|y_2\|(x_2, y_2)) / (\|y_1\| + \|y_2\|)$.



(a) The ADF (all links are ordered pairs) **(b)** Main Program (all links are scalars)

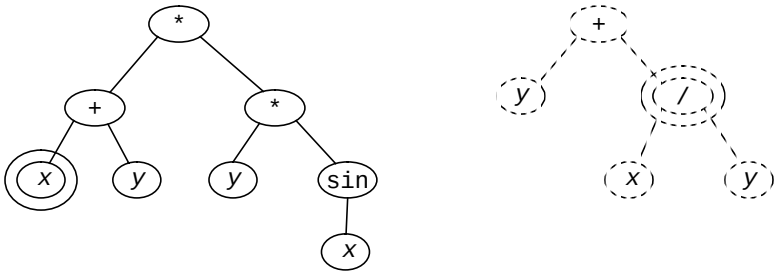
Figure 11: An example individual with ADFs. The left side shows the ADF. `prev depth` is the ordered pair of distance estimate with confidence. The program takes a weighted average of the first estimate with the best of the other two, returning an ordered pair of depth and confidence. In the main program, these values appear as two scalar terminals. The main program modifies this estimate (in this example, the modification is meaningless). In general there can be many ADFs, and the main program can perform arbitrary post processing.

Figure 9, plus a number of constants. We start by creating a number (say 50) trees at random; this first set is called "generation 0." We then evaluate each one and assign it a real valued score. We then choose a parent or two at random, but trees with higher scores get higher probabilities. Specialized genetic operators are used which generalize sexual recombination and mutation for the tree structured computer programs undergoing adaptation (see the figure). After generating 50 children this way, we label them "generation 1," evaluate them and repeat.

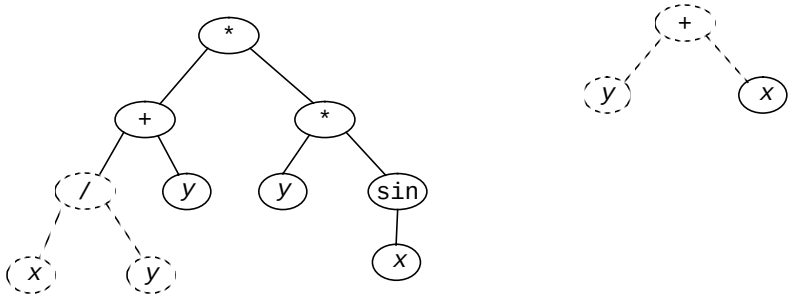
We're trying to estimate both depth and confidence, which no doubt need to be estimated together. As such, it seems reasonable to include "ordered pair" as a data type, along with "real number". A function which split an ordered pair into two real numbers, i.e. that has one ordered pair input and two real outputs, destroys the tree nature of the program, making crossover problematic. A function that returns only one number of the pair discards all the work on the other, since there's no other way for that value to be used. As a result, I propose to use *automatically defined functions* (ADFs) [Koza 1994, *Genetic Programming II*], and limit ordered pairs to them.

An *automatically defined function* is a function, or subroutine, or procedure, or module, or program, that is evolved during a fun of genetic programming and which may be called by the main program that is being simultaneously evolved during the same run. Each individual consists of one main program and one or more other programs (the ADFs). In the main function, a call to an ADF appears just like any other function (if it takes arguments) or terminal (if it doesn't).

For the thesis work, the idea is to have the ADFs output ordered pairs. Instead of appearing as a single ordered-pair-valued terminal, these outputs will take the form of two separate real valued terminals. See the example in Figure 11.



(a) Two individuals before crossover, with the crossover points marked ...



(b) ... and after crossover

Figure 10: Genetic Programming. In genetic programming, programs are represented as parse trees. For example, the tree in the upper left represents the expression (*x* + *y*) * *y* * sin(*x*). Mutation is accomplished by selecting a node at random, deleting the subtree rooted at that node, and inserting a randomly generated one. Crossover is implemented by picking a random node in each parent, and swapping the subtrees rooted at those nodes.

Table 2: Functions to be used with Genetic Programming

Function	Description
+, -, *	Ordinary arithmetic functions (both binary and unary -).
%	Protected Division (binary). Does normal division, unless the denominator is zero; in that case, it returns 1.0.
mod	Real numbered modulo (binary). If its second argument is zero, it returns zero, otherwise it returns x mod abs(y).
abs	Absolute Value (unary).
srt	Protected Square Root (unary). Returns sqrt(abs(x)).
sin, tan, exp, rlog	Basic transcendental functions (unary). If the argument to rlog is 0.0, rlog returns 0.0. Otherwise, it returns log(abs(x)).
gt	Greater Than (binary). This is a real valued version, using +1 for true and -1 for false.
ifneg	If Negative (trinary). If the first arg is negative it returns its second, else it returns its third.

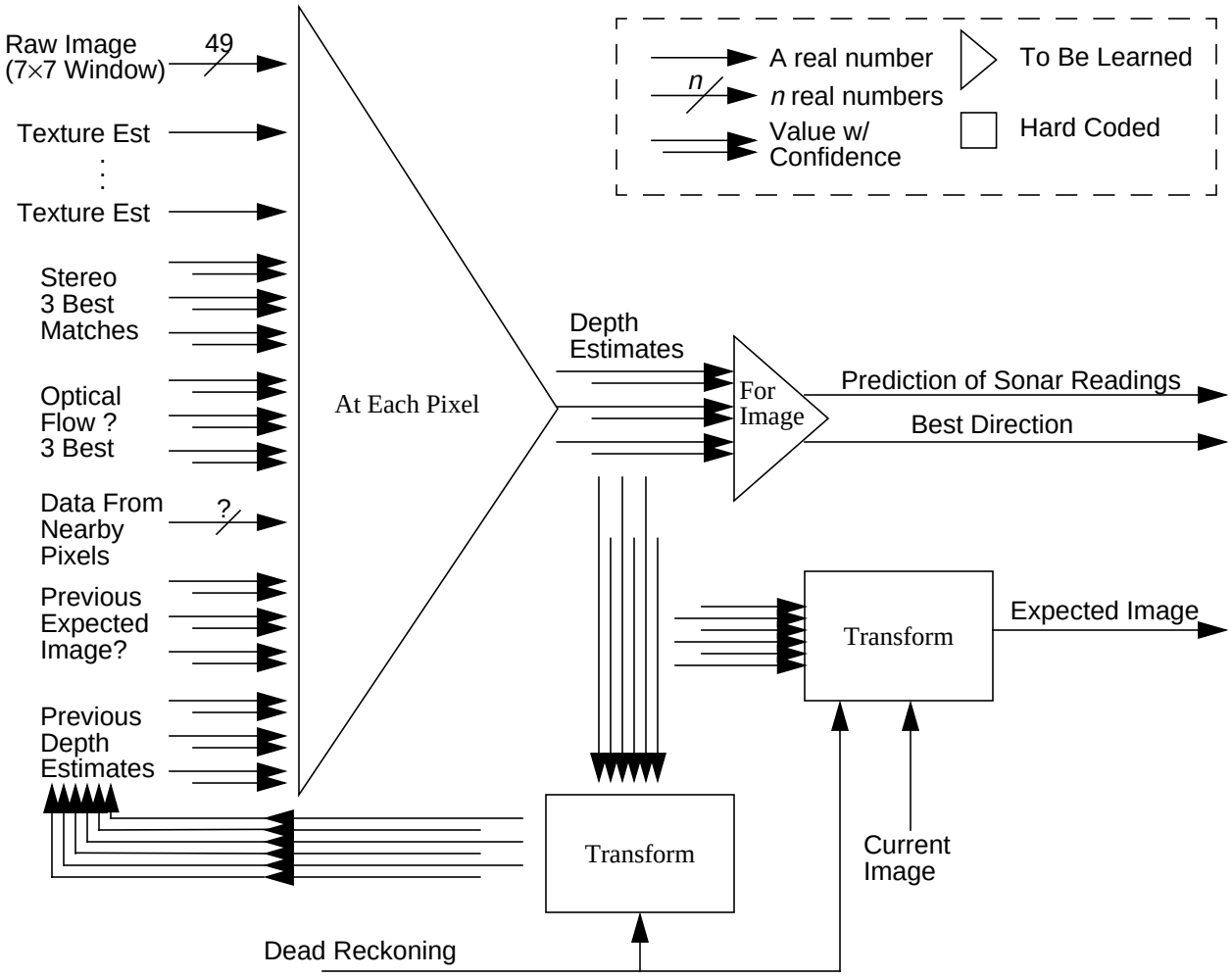


Figure 9: Structure of the perceptual system. This figure corresponds to the shaded box in Figure 8. The incomming image is processed to give a number of different depth cues. Depth is estimated on an 8 by 8 pixel lattice. At each pixel, the same function is evaluated (large triangle). It takes the image, the depth cues, some subset of that from neighbouring pixels, and the depth expected at this pixel, based on the depth map estimated at the last time step, transformed according to dead reckoning. The resulting depth estimates are collected in an array and fed to a second program, which predicts the sonar readings and calculates the best direction for the robot to travel. Both these functions are learned. Other hand coded functions compute the transformations based on dead reckoning.

sents a function as its parse tree (see Figure 10). Somewhat unexpectedly, genetic programming (and evolutionary computation in general) have been shown to be efficient enough for practical use [Kinnear 1994, *A Perspective on the Work in this Book*]. One can conceptualize the solution of a problem as a search through the space of all potential solutions to that problem. Taking this view, genetic programming is considerably more powerful than alternative search techniques such as exhaustive or random search. It implicitly utilizes a directed search, allowing it to search the space of possible computer structures and find solutions in hours on tasks that would take random search considerably longer than the life of the universe. While computationally still challenging, genetic programming can produce useful and dramatic results in minutes and hours instead of millennia.

The set of possible *function* (i.e. non leaf) nodes, as well as the set of possible *terminal* (i.e. leaf) nodes, is set by the programmer ahead of time. For the thesis work, the list of functions is shown in Table 2 on page 24; note that none of them have side effects. The set of terminal nodes are the inputs described in

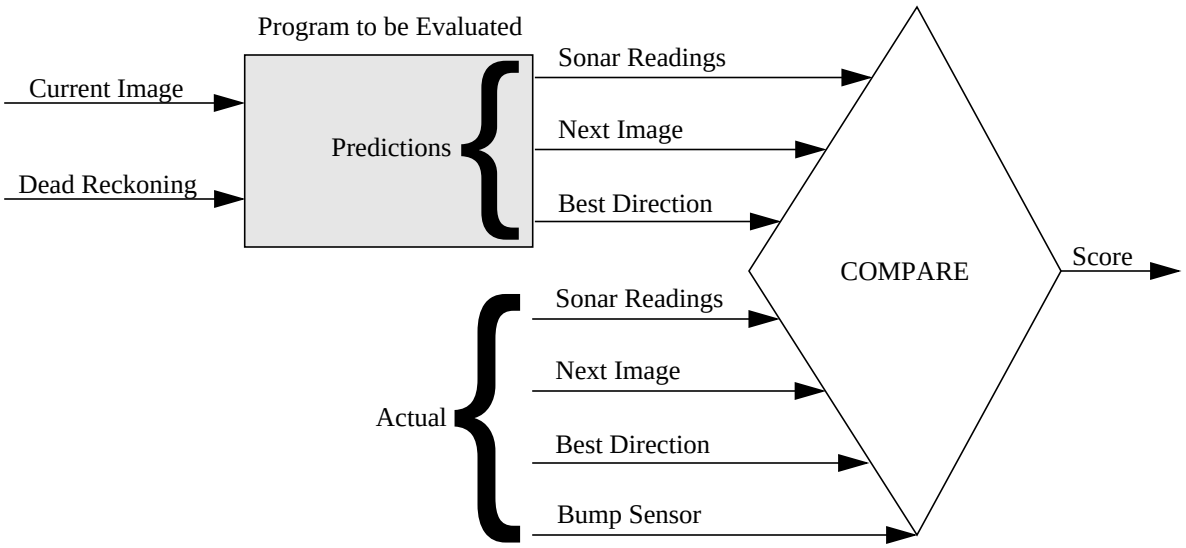


Figure 8: How potential programs will be evaluated. Note that the program will have it's own state. At each timestep, the robot will take an image; the program will try to predict the current sonar readings, the next image and the best direction to travel. These are compared to the actual values, the robot is moved in the predicted best direction, and the bump sensor is monitored. If the robot hits an obstacle, the run ends immediately. Otherwise, the process starts over, until a maximum number of timesteps is reached.

hope, however. For example, textureless areas usually represent a smooth change of depth, since discontinuities (e.g. occlusion boundaries) and non-differentiabilities (e.g. edges) tend to produce lines.

These considerations have lead me to the following framework (see Figure 9). For each pixel where a depth value is needed, we compute three possible depths, each with it's own confidence. We process the following inputs:

- A small window (e.g. 7x7 pixels) of the raw image, centered at the current pixel,
- A fixed number of "texture estimates". These are produced by simply convolving fixed masks with the raw image, around the point of interest. Candidates include lines oriented at various angles, Gaussians, derivatives of Gaussians, sine / cosine waves, etc.
- The three best stereo estimates, with confidences,
- Perhaps the three best optical flow estimates, with confidences,
- Some subset of this information for windows neighbouring the current point. The previous depth estimates are essential (this is how depth info can be spread to textureless areas); what else to include will be decided by experiment.
- The three depth values (with confidence) estimated at the previous time step,

In this framework, what we seek is a mapping from some real numbers (the set of inputs listed above) to other real numbers (the three depth and confidence estimates). Our problem is to extrapolate from the training data to new data. There are a number of function approximation frameworks for the task, but most assume some sort of continuity, so that similar inputs have similar outputs. Neural nets, Locally Weighted Regression and Bayes Classifiers all fall in this category. Yet, such techniques can't represent many candidates, at least not easily. For example, computing the rule "select the stereo estimate with the highest confidence" (`argmax`) is awkward, as is "use stereo where there's lots of texture and the previous depth estimate otherwise". These involve selecting from alternatives and other discontinuous functions.

A technique which accommodates such discontinuities naturally is genetic programming or GP [Koza 1992, *Genetic Programming*]. Genetic programming is an offshoot of genetic algorithms which repre-

can make full use of a cluster of workstations. Finally, it doesn't require the constant supervision and resetting by hand that experiments with real robots do. However, the prevailing opinion is that computer graphics isn't up to the problem of reproducing the peculiarities of CCD cameras. That is, techniques that work on simulated images are likely to fail on real images.

Another possibility is to create a simulation as above, but replace computer vision techniques with a collection of images from positions on a tightly spaced grid, under a number of different conditions, and then "interpolate" between them. If we want an image from a certain location, the interpolation can be done, for example, by projecting the nearest image onto the scene, then projecting it back onto the simulated camera.

However, both these approaches have the problem that simulations necessarily embody assumptions. For example, we might add random, gaussian noise to our dead reckoned position to account for error. But such noise is actually very well behaved. For example, if each move has k percent error, then after n moves our error is only $k/\sqrt{n}$ percent. That's in marked contrast to a constant error, which can accumulate very quickly. Also, such errors may have different characteristics on carpet compared to concrete. There may be ways to overcome these problems (like exploring the characteristics of the dead reckoning error and produce a better model of it), but I don't want to stake my thesis on it.

And so I want to evaluate my programs by running them on the real robot. Given this, how can the robot automatically sense how well it's avoiding obstacles? If we augmented the robot with bump sensors we'd know when we hit an object, although we'd have no way to detect drop-offs. But that one bit of knowledge at each timestep is a poverty of information; discovering subtle patterns in a digitized image, which is just a big array of numbers, would be pretty much impossible.

Another approach would be to set up some external positioning system, such as beacons or retroreflective strips, and type in a map of all the obstacles in the environment. Besides the time and expense involved in setting up this system, it's very time consuming to move to other buildings, and it can't know the location of dynamic objects, such as people or even empty water cooler jugs.

Sonar provides more information, although it too fails to detect drop offs. Given an image, the trial program could guess the sonar readings and the best direction to travel. In other words, we could use sonar as a standard bearer or baseline. We could in this way achieve at least the competency of sonar based obstacle avoidance. Sonar isn't perfect—it's noisy, it can only resolve to within 30 , and suffers from problems like specular reflection—but it looks like the best option available.

So, in order to assign a competence score to potential programs, we can have our program predict the sonar readings, as well as predict the correct direction to travel. If the robot collides with an obstacle before the end of a run, a bump sensor can detect this and the program can occur a penalty that decreases as time goes on. A final type of feedback, which is potentially the most useful, is to have the robot try to predict what the camera image will look like at the next timestep. To succeed, the robot needs to understand how objects are moving with respect to itself, in a way that should help with obstacle avoidance. See Figure 8.

Now that we have an overview of the robot and how we intend to "score" potential programs, let's look at a framework for perception and control.

The Representation of Learned Programs

One of our goals is to integrate a number of different depth cues. However, having the robot discover the depth cues on it's own is probably too ambitious. So as input (see Figure 9), the perceptual system will get the raw image, the top 3 stereo matches (with confidences) every 8 pixels, and similarly for optical flow, the top 3 matches (with confidences) every 8 pixels. We also include some measures of texture and gradient magnitude. To allow for expectations, we'll provide as input the depth map generated at the last timestep, appropriately transformed according to dead reckoning, to give the expected depth at each pixel. The learning algorithm will discover on its own how to integrate all these cues to produce a new depth map, and how to estimate the sonar readings and the next image.

One of the big problems with methods like stereo vision and optical flow is that they can't handle textureless areas, such as the plain grey walls around my office. This is because they are *local* methods: they calculate the depth of the area based on images of that area alone. It's not hard to see that an array of plain grey pixels could be at any depth, and any subset of it matches well with any other subset. As a result, the depth of one area needs to depend on the depth of surrounding areas in non trivial ways. There is some

Table 1: Possible sources of feedback information for learning

	Pros	Cons
Total Simulation	• Control of environment • Can simulate dead reckoning error, specular reflection of lights, sensor and actuator error • Speed bottleneck: CPU (arbitrarily fast) • No human intervention needed during run (can run overnight)	• Necessarily embodies assumptions • Hard or impossible to model all significant problems of imaging. This means what works in simulation probably won't work in the real world.
Simulation w/ Real Images	• All advantages of "Total Simulation" • Most or all problems of imaging are represented	• Necessarily embodies assumptions • Need to collect a dense set of images
Bump Sensors Only	• Doesn't embody assumptions as to what's good and bad for the computer to do	• Can't detect drop offs • Only a few bits of feedback per run
External Positioning w/ Handmade Map	• A lot of feedback per time slice (e.g. distance to nearest obstacle) • Could know location of drop offs	• Large time and expense to set up • Doesn't move easily to other test environments • Doesn't handle dynamic obstacles
Map from Sonar (w/ Bump Sensors)	• Easy to set up/Portable • Handle's dynamic objects	• Can't detect drop offs • Sonar is noisy
Sonar, Bump Sensors and Predict Next Image	• Easy to set up/Portable • A lot of detailed feedback • Needs to understand a lot about objects in the environment to succeed	• Doesn't work on dynamic objects • Complex/time consuming to program • Needs to understand a lot about objects in the environment to succeed

this area. The most problematic visual features are the glossy grey walls which often confound local depth estimation techniques such as stereo, optical flow and depth from focus.

The Evaluation Method

Since we wish to use a learning technique to generate the software, let's focus on how to evaluate potential programs (See Table 1 on page 20). Above I mentioned the possibility of using simulation to evaluate various software designs. We could create a simulated world, and whenever we need a camera image, we could use computer graphics techniques to render an image. When we command the robot to move, we update it's position in the simulated world, complete with error: we move it to a slightly different position than was commanded.

This use of simulation is attractive for a number of reasons. First, recent ray tracing and radiosity techniques can simulate many of the problems of real images, such as specular reflection of fluorescent light bulbs on glossy paint. This technique also allows a large number of candidates to be evaluated quickly, and

6. After some investigation, we have concluded that the cause is almost certainly a mechanical problem with the rollers, either slipping on the ground or moving slightly up and down their axis. The problem defies a simple fix in software, since the error depends on direction of travel and whether we're travelling over carpet or hard floor.

Instead of running each program on a physical robot, we can simulate it in software. This alone can speed things up many orders of magnitude, especially when clusters of computers are used. However, even this can't reduce the time needed by such a large factor (more than a billion to one).

The hope comes from the fact that while species have been evolving, so has evolution itself. For example, the advent of sexual reproduction made evolution much more efficient, speeding up the pace of new developments. Also, gene expression—the mapping between a genome and the organism it codes for—is crucial to the efficiency of evolutionary computation. If the mapping took similar genes to radically different designs, then the genetic operators (crossover, mutation, etc.) would have essentially random results. Evolutionary computation is hopeless at solving cryptograms or inverting hash functions.[5] But if the effects of genetic operations produce semantically reasonable results most of the time, then the search is far from random and we have some hope. Evolution has been working within a certain framework since the beginning (always with DNA, and the first stages of gene expression haven't changed significantly since evolution started), so we should not be surprised if there is a lot of room for improvement.

7. The Thesis Implementation

The Robot

Let's look at a specific application: visual obstacle avoidance using the Uranus mobile robot. Before discussing the details of the learning, it is helpful to discuss the practicalities of the robot and how to evaluate potential perception and control programs. The robot to be used in this thesis is the indoor mobile robot Uranus, the mascot of the Mobile Robot Lab (Figure 7). It has a 3 degree of freedom base with resolvers on all four motors for dead reckoned positioning. When driven forward or turning in place, the dead reckoned position is fairly accurate (about 1% error), but when driven sideways the accuracy drops substantially (10 - 20% error, significant rotation)[6]. For sensors it has three monochrome cameras which we plan to upgrade to color cameras soon. It also has a Denning sonar ring of 24 sensors, and we may add bump sensors to it.

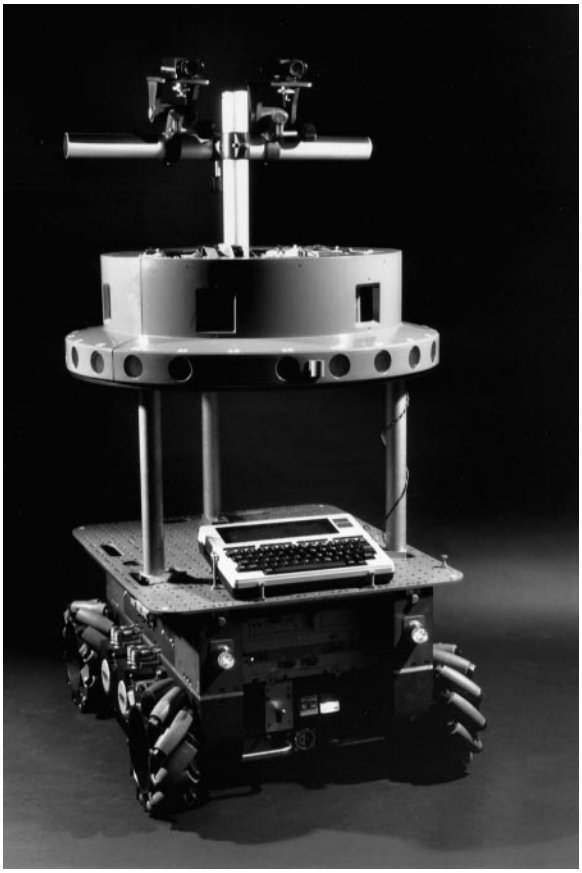


Figure 7: The Uranus mobile robot.

For computing it has an onboard 68000 based computer running VxWorks, which communicates over a VME bus with a motion controller board. The 68000 board accepts commands of the form "go to *x*, *y*, *theta*" or "travel at speed v_x, v_y, v_{theta}," communicates with the sonar hardware and also integrates the dead reckoned position of the robot. It communicates with a host computer over either ethernet or an RS232 serial line. The host computer can be any desktop computer; at the moment it's a Sparc 5, but soon we hope to upgrade to a dual Pentium II running OS/2 (and perhaps later BeOS).

The robot is largely restricted to the hallway and offices near my office, due to its weight (it takes two people to lift it) and the lack of elevators in the Field and Mobile Robotics Building. Although data collection and testing in other buildings is important, much development and testing will of necessity take place in

---

5. I am assuming the "obvious" technique: use the plain text as our genome, the encryption of it as our individual, and some straightforward distance metric in the space of encrypted text as our fitness function. Some other, more subtle use of evolutionary computation may be very applicable to such problems.

One final point. Although Brooks' point about representation is a good one, it doesn't by itself lead to subsumption. It's quite possible to build systems both without much representation, and without strong separation.

Summary

The architecture of robot software is designed by hand. To manage complexity, such architectures are invariably composed of a number of black boxes. Even the alternative approaches above, at some point, break the software down into a number of conceptually independent subsystems with clean interfaces, and with the peculiarities of their implementations hidden from each other. This blocks the development of systems with the subtlety (and resultant ability) of the human perceptual system, as described in Section 3. So, how can we break out of the black box?

6. A New Approach

What is the alternative to black box perception? A black box structure with strong separation appears necessary for managing complexity when people design systems. Therefore, we need to come up with some technique for designing systems that are more interdependent than what people can design. In other words, we need to hand at least part of the design process over to software tools. Our problem then becomes the meta problem: how do we design tools that design robot software?

What could those tools be? How can they design fundamentally, qualitatively different systems, on par with nature? Since nature has been so successful, let's take a page from its approach to design. The fundamental difference is that nature starts with a number of designs, chooses the ones that work best, and combines them in hopes of getting the best of both or even something new. In other words, it uses learning[4].

Although learning is by no means new to robotics, it isn't usually applied to designing architectures. As with connectionism (see the previous section), most applications of learning start with a decomposition into subsystems. This usually done by hand, in a black box way, and only the internals of the boxes are learned.

If evolutionary computation or some other learning technique is used to create the design, the subsystems can be tuned to the particular task and environment. Whereas a technique or theory of vision strives to be generally applicable, a technique or theory of vision *design* can be general, while the designs it creates are specialized. This also allows the individual subsystems to be more interdependent. Besides not having to be "plug and play" in other designs, they are designed together, so they can learn to compensate for each others weaknesses. And the design doesn't need to be simple enough to be understandable by the programmer at all levels. This isn't to say that anything goes or that it can be arbitrarily complex. As pointed out above, animals have a lot of structure. There is only a small number of basic cell types, for example, and different functions are located in different areas of the brain. But the separation between parts can be a soft separation.

Above I argued that hard constraints work most of the time, but fail every once in a while. This makes them perfect for a learning algorithm. Hopefully, a simple, general rule will be easy to find. Once found, it is likely to be kept because of it's great boost to performance. Then, the cases where it fails will be noted and exceptions made for them. This kind of fitness landscape, where there is a nearby approximate solution that can be further refined, is the most conducive to learning.

Before heading to the specifics of my application, let me address a common objection to the use of evolutionary computation in particular, and unstructured learning in general, for such design. Nature isn't really an example of a practical design system, since it took billions of years to evolve the level of competence we're talking about. No one wants to wait a billion years to evolve a robot. In order to make the suggestion more palatable, I need to provide some hope that evolutionary computation could produce a practical design in significantly less than a billion years.

---

4. This is learning in the computer science sense, not the biological or psychological sense.

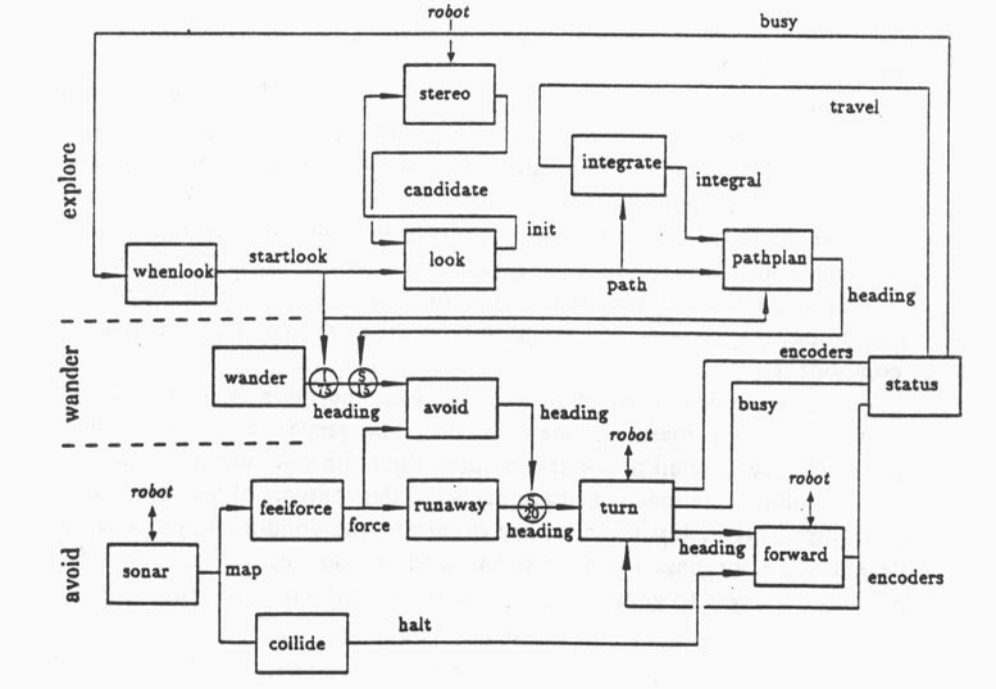


Figure 6: The architecture of a typical subsumption robot. Note that, although there are 4 or 5 connections between layers, each box has only a few inputs and outputs, and the errors of one box are ignored by the other boxes. From Brooks 1991, *Intelligence without representation.*

perimentation and knowledge moves from the task domain itself towards a metalevel, of how to automatically create such programs. This is an important point to keep in mind the next section.

Subsumption

Rodney Brooks' main criticism of robotics is that representation is overused, and as one alternative he suggests the subsumption architecture. However, the use of representation and the use of black boxes are largely independent. In subsumption, as with traditional approaches, the architecture of the system is designed by hand, and to manage complexity the system is broken into a number of black boxes (see Figure 6). The boxes are organized into layers, each layer designed for a separate task. This allows task knowledge to come into play, but in practice the task knowledge is embodied as a hard constraint.

The "hard constraint" thesis might be that the constraints of the real world can be described simply (e.g. "the floor is textureless, while object boundaries aren't"). The failure modes come from exceptions, which always seem to exist (e.g. bright bands of sunlight, patterned carpet). This doesn't mean that such constraints should be ignored; rather, it should be noted that they help most of the time, but there are occasional exceptions. It's like conceptually driven perception: often the stimulus is unambiguous, but when it isn't, you may be better off using other constraints (such as expectations, semantics, etc.).

The black boxes are often at a lower level than in the traditional approach, but they still have very circumscribed interactions, and implementation details are hidden within each box. Also, the interactions between layers are largely in one direction. While higher levels can "read" any lower level output, the canonical way for higher layers to *affect* lower layers is to "subsume" an output, that is, to replace it wholesale, "tricking" the lower level inputs.

The reason for black boxes (and a hierarchy of black boxes) is to manage complexity, and makes large projects and ambitious tasks possible. By using a black box description only at a lower level, the subsumption systems built so far have performed comparatively simple tasks. However, this may be because the number of such systems built to date is relatively small. The field is still in its infancy. Still, the question of how the subsumption architecture scales is open.

about any of the issues of how they're designed. Although this is a great win from the complexity and division of labour standpoint, it makes for a lot of isolation between subsystems. I call this type of isolation *hard separation*.

In contrast, to get the most information out of a sensor's readings, our perception system is *conceptually driven:* task and domain knowledge influence even the lowest levels of perception, and the different levels influence each other. However, a conceptually driven perceptual system, by definition, can't have hard separation between its components. This is not to say it must be one undifferentiated mass; there should no doubt be identifiable subsystems, but these subsystems need to interact in subtle ways. I term this type of interaction *soft separation*.

Marr and Nishihara reflect the traditional view in all of robotics when they take hard separation as the fundamental tenant of computer vision. In their view, research in computer vision should not worry too much about the issues in other fields, but instead focus on finding universal constraints for this underconstrained problem. They naturally pick only a single set of related issues to study, which they can study in great depth. And they search for techniques that are as generally applicable as possible. In this way they are lured into creating black boxes that are hard separated from other field's boxes, and from task and domain constraints. In fact, it's hard to see how else the problem could be attacked. In Section 6 I explore some general alternatives, but first let's examine some existing, non-traditional paradigms in light of this critique.

5. A Critique of Alternative Approaches

There are a few new approaches on the market, such as connectionism and subsumption, that are outside "traditional approaches to robotics." It isn't clear at first how the above critique applies to them, and so they deserve a few words of their own.

Connectionism

Since a connectionist network is (typically) densely connected, it's hard to decompose it in terms of subsystems. In fact, it's not even clear that there *are* separate subsystems. This would obviously circumvent the problem of hard separation. However, neural nets are rarely used for high level tasks, such as planning or even obstacle avoidance. For one thing, the data structures used by neural nets are limited to vectors of real values. It's not clear how to represent plans, for example, in this notation.

For this and other reasons, a single neural net isn't typically used for a high level task such as path planning. Instead, the task is broken down into subsystems by hand, with the programmer deciding the overall structure, as well as the input and output semantics of each subsystem. But by this point we already have a black box structure. The neural net is used strictly to implement the black box.

Neural nets are typically feed forward, and so can't make use of expectations from previous time steps. For example, a neural net based vision system typically treats each image independently. This amounts to treating each image as if it were the first, and not trying to verify previous interpretations. However, neural nets do make use of domain and task knowledge. Whatever regularities are part of the training data can make it into the network. They can't, however, do this in a cross subsystem manner; in the case if the ambiguous "A" and "H", a neural net that analyses each letter couldn't use the identity of the word to help disambiguate individual letters.

A particularly relevant example is ALVINN [Jochem 1997, *The ALVINN Project Home Page*, and Pomerleau 1993, *Neural Network Perception for Mobile Robot Guidance*]. It's goal is relatively low level: to classify the scene as to a gross property of the road, namely the direction it's turning (ALVINN didn't do any obstacle avoidance, only navigation). Note that this problem is well suited to function approximation. Small changes in the inputs result in small changes in the outputs, and small errors in the outputs can be corrected at the next iteration. By using a single network to map inputs directly to outputs, there are no separate subsystems and many of the problems above are avoided. About the only "missed opportunity" is not reusing the results from the previous timestep.

But what's more, creating ALVINN entails a shift of thinking. The designer doesn't need to know the issues in finding road direction; instead, they need to know the issues in neural nets. The expertise, ex-

However, in this work I'm interested in its impact on robotics alone. Any larger scope would have to be the subject of another paper.

A Final Example: Depth Cues

Let's see how this plays out in the case of using multiple depth cues. In Section 3 we noted that people combine a number of depth cues, and it's unclear whether a single depth cue is sufficient for even simple tasks such as obstacle avoidance. Certainly researchers recognize the benefits of combining depth cues. In fact, it can be considered an example of the larger problem of sensor fusion. Why aren't more people trying it?

Again, the problem is isolation and complexity. Traditionally, for each depth cue, we analyze an image and return a depth map. It might make more sense to return a set of constraints, but what language would we use to describe them? We could also try combining the depth maps, but again how do we do that? Answers like "take the average" or "use the most common" are typical data driven answers, because they're optimal assuming that the errors of each method are independent of each other, independent from pixel to pixel, independent from one moment to the next, independent of the geometrical arrangement of objects in the environment, and in general independent of all other variables in the robot, it's environment and the task.

Formalization

As part of the thesis work, I intend to formalize the definition of separation, by producing a metric for the amount of separation. With such a definition, existing robot designs could be compared in terms of separation of their components, and tested on noisy, real world data, to see if the softer separated robots work better. The definition could also be used as an analysis tool: in the context of a particular robot, the formula may suggest ways to weaken separation, which may lead to novel innovations.

Some initial issues and ideas are as follows:

- Each box transforms its inputs to its outputs. A certain amount of information is lost in this process; hopefully the irrelevant information is lost and what's essential is retained.
- Information theoretic concepts may help to formalize this. For example, the perceptual system as a whole wants to maximize P(`correct interpretation|inputs`). If a certain box transforms raw sensor inputs alone to an intermediate representation "`rep`," then P(`correct interpretation|rep`) will be lower than the original probability. We could analyze the theoretically optimal performance of this processing step by determining how much lower the second probability is. If too much is lost, we may conclude that some form of feedback or task or domain specialization is necessary.
- This could even be used to judge the adequacy of a representation. For example, people are quite good at recognizing scenes from edges alone (see any line drawing, such as a black and white cartoon). Therefore, there is a transformation from image to edges that preserves the relevant information, so using edges as an intermediate representation is at least theoretically possible
- One way for a box to not lose any information is if it does nothing. Therefore, we can define the work done by each box as being the information lost by that box, namely P(`inputs|rep`). The best boxes are the ones that do a lot of work but don't lose a lot of information. It is hoped that softly separated systems are better at throwing out the irrelevant while retaining the relevant.
- The above analysis would help formalize and explore when soft separation is better than hard separation; a more formal definition of soft separation would help as well.
- We also need a way to analyze boxes that maintain state and use expectations, and task and domain knowledge.

Summary

The problems and limitations of current robot perception algorithms come from using the traditional method of managing complexity: the *black box* methodological stance. This stance, which has been wildly successful in science and engineering, works by isolating subsystems so that the description of what they do and how to use them is much simpler than how they do it. Also, those who use them don't need to know

original description and work around it. To keep the description simple, we usually give a superset of the error conditions (e.g. "during an `fdiv` statement", even though only certain numbers caused problems), and a bound on the error (e.g. produces accuracies as bad as four decimal digits). If this isn't possible, we throw it out and get a new one. When we design and build systems using a black box, all we use is its description, possibly updated in the face of such errors. If an error is discovered in one subsystem, all bets are off and in general no one knows what the system as a whole will do. This is why people wanted new Pentiums; they had no idea what the ramifications of the fdiv bug were for the software they were using, and for software makers to prove that errors would be small would be a huge task. The simplification accomplished by black box decomposition is huge; even a small deviation from the spec leads to a great ballooning of complexity.

The Interaction/Complexity Trade-off

Comparing this with the conceptually driven nature of human perception above, the difference is apparent. Those who are interested in how words get their meaning are in a separate group from those who study the shape of letters and optics, so they create separate subsystems. For example, to make them usable by each other, the latter might create programs that use camera images as input and ASCII as output, and the former might accept ASCII as input. Once the first stage decides on a word's identity, that choice is never revisited.

But taken as a description of the field, it's is oversimplified. The black box framework doesn't *require* that the intermediate representation be a single ASCII character for each letter seen. Instead, for example, it could return a list of letters with associated probabilities. For the character between A and H, it could return A and H, each with a probability of 0.5. In general, a black box in a perception program could return a "short list" of possibilities with associated probabilities.

But even so, this makes for slower perception. It can't account for the use of semantic expectations to speed up recognition (see the "nurse-doctor" "bread-doctor" example above). More importantly, it means that errors can't straddle black box boundaries. The system can never parse a situation where the physical stimulus is not all that close and neither is the syntax, but together there's only one consistent solution. In other words, if each subsystem only provides a weak constraint—and in general, enumerating all the possible interpretations that satisfy the constraint would be far too gigantic a task—the "list of possibilities" representation is impractical.

Hard Separation vs. Soft Separation

The next fall back position is for each black box to provide constraints in some constraint language. But the most natural language for such constraints use the concepts from the analysis at that stage—the very concepts that need to be contained within the black box, to manage complexity. More importantly, any nontrivial analysis reduces information. And without knowledge of syntactic, semantic, task and domain constraints, we can't be sure what's relevant and what we can safely throw out.

In other words, what characterizes the black box approach is *hard separation*. Since each box performs a qualitatively different task and has a simplified external description, there simply isn't much room for subtle interaction between two boxes. Once we've decided on the structure of the system—how we'll break it down into subsystems—the subsystems become fairly isolated. As a result, the structure we impose is rather rigid.

This is not to say that there should be no separation, or that there is no separation in the human brain. Certainly the human perceptual system is composed of a number of subsystems, physically located in different parts of the brain. However, these subsystems work fundamentally more closely than the subsystems in traditional robots. Using syntactic and semantic constraints in the word recognition subsystem is not possible in a system with hard separation; I refer to such interplay as *soft separation*.

It should be noted that the hard separation / black box approach applies not only robotics, but to most of Western science and engineering, if not other avenues of thought. We build our machines to have predictable and easily describable behaviour. The connotations of the word "mechanical" come from this. Anyone who has worked with machines knows that naturally they shake and bend, that electronics is susceptible to noise and even chaos, and that a lot of effort is expended on finding rigid materials, precision machining and linear amplifiers. [Quote the GEB line about "why can't machines invoke the imagery of energy flowing from place to place?"] Viewing other endeavours in this light may lead to fruitful insights.

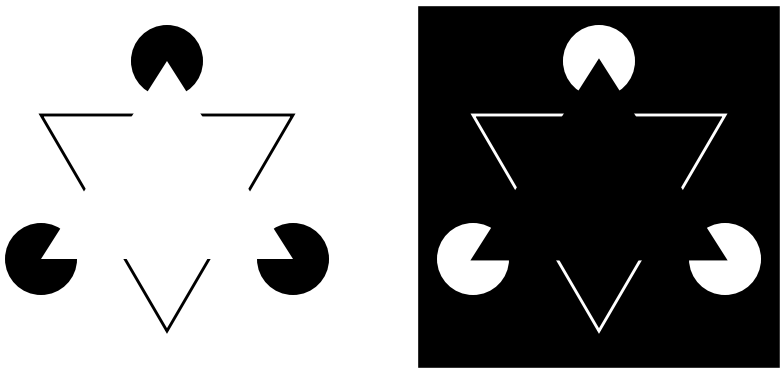


Figure 5: Subjective contours. On the left, we see a white triangle whose verticies lie on top of the three black circles. The three sides of this white triangle (which looks brighter than the white background) are clearly visible, even though they don't exist physically. On the right, black and white are reversed. Here, there is a black triangle (which looks blacker than the black background) with subjective black contours. [Kanizsa, 1976]

cles, we use bottom up processing because it's all we have left. But in more normal circumstances there is a ton of extra information we bring to bear. Whereas Marr and Nishihara think in terms of the vision problem being under constrained and what constraints to add to it, it seems more helpful to look at it as "over suggested," and ask how we can best take all our information into account.

As we've seen above, obstacle avoidance based on bottom up perception doesn't work reliably for long periods of time. Even those researchers searching for constraints recognize the potential benefits of including more conceptually driven elements in perception, yet still choose not to investigate that path. The main reason is complexity; such a system would be considerably more complex than a data driven pipeline system. To manage complexity, they decompose intelligence into a number of smaller, independent problems (e.g. find edges given only the image), which are individually tractable. The particulars of how scientists manage complexity shed light on both the limitations of current robotics and the possibilities of circumventing them.

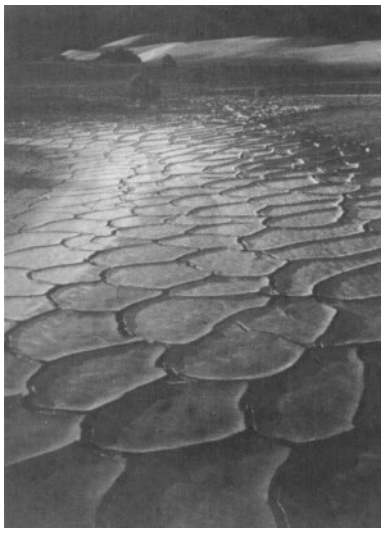
Black Box Systems

There are many concessions to complexity that researchers make, but two in particular concern us here. The first results from specialization, in which each researcher focuses on a different subsystem of a robot. Examples of specialties include perception, planning and position estimation. Robots, after all, are complex systems and by specializing in how to model, analyze and build in one area, researchers can explore that area in greater depth. But at the same time, researchers want their approach to be usable by many different people, in many different areas. So they make them as independent of the other subsystems as possible, and usable for as many tasks as possible. Thus, robot software is composed of a number of black boxes.

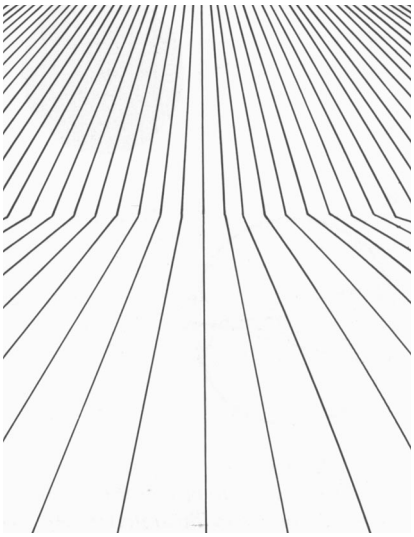
In this paper, I intend the phrase "black box" to be a technical term to describe the components of such a design. The essence of a black box is that the task it performs and how it is used can be described succinctly, and that as long as one has this description, there's no need to know how it works.

A few clarifications are in order. To say that its function and use can be described succinctly, I mean that in order for someone outside that subfield to use it, they don't need to know as much as the implementers did, let alone the designers. For example, a canonical black box is the microprocessor. Although microprocessors are quite complex, those who design and build PCs don't need to understand this complexity; all they need to understand is the interface: the power pins, the address, data and control pins, and so on. And the interface to a black box is often described using new concepts or abstractions. Of fundamental importance to the microprocessor are the concepts of the *instruction*, the *operand*, various *addressing modes* and the like. Together these form the abstraction of the machine language.

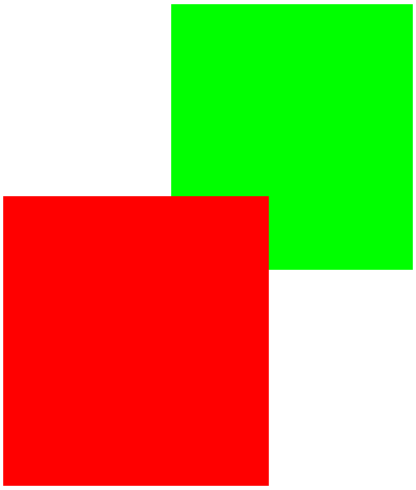
When a black box doesn't conform to its interface, we declare it broken (e.g. the Pentium bugs). If we can describe under what conditions the error happens and what goes wrong, we can roll that into the



(A) Texture gradient Uniformly textured surfaces produce texture gradients that provide information about depth, for example, in the mudflats of Death Valley.

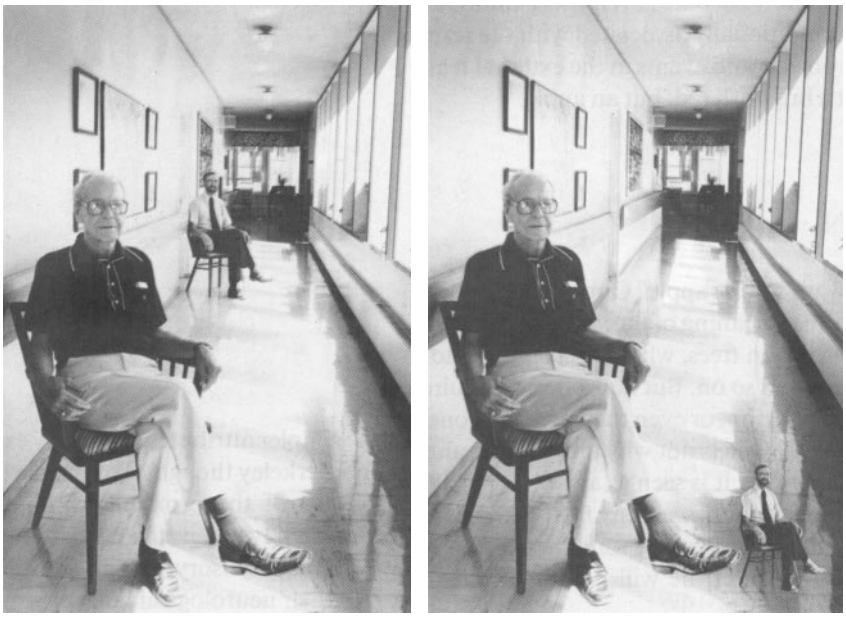


(B) Changes in texture gradients Such changes can be cues to corners, occlusions and other phenomena.



(C) Occlusion

(D) Linear perspective



(E) Perceived size and distance The 2D size of the men in the image—which corresponds to the size of their retinal image—is in the ratio of 3 to 1. But in the left image, they look roughly equal in size, with one about three times further off than the other. In the right image, however, the smaller man appears to be at the same distance as the other man, and one third the size.

Figure 4: An assortment of depth cues.

> tic knowledge can specify fairly accurately what kind of word must come next—syntactically, it will probably be a noun or an adjective, semantically it must be some physical thing of enough value to be worth stealing. Thus context, both of syntactic and semantic nature, is having an influence on the process of word recognition. Indeed, such context is even influencing the phonological component, in that the entire phonological representation of a word is not necessary to identify and comprehend the word. [Ashcraft 1989, p. 429]

Integration of Depth Cues

A final point about depth cues. People don't use a single cue to determine depth, but integrate a number of cues. These include binocular stereo, texture gradients, linear perspective, relative size, occlusion, focus, support and optical flow; some examples are show in Figure 4. While all of these are individually fallible, together they work pretty well.

There is often a tendency, among roboticists, to consider vision synonymous with stereo, so let me point out that in human perception there are many times when stereo isn't even used. For example, if we cover one eye we can still tell which objects are close to us and which far away. And when we watch movies, stereo isn't giving us no information, it's giving us wrong information with complete certainty. Stereo tells us that everything is at the same depth (that of the screen), yet we can still perceive majestic landscapes stretching off into the distance, or that Tyranosaurous Rex lunging at the Brontosaurous. Even when we're looking at a still image (which lacks any motion cues), we have a good idea of how far away the different parts of the scene are. [Show some visual illusions? Point out that illusions usually involve removing as many depth cues as possible? It's hard to find natural illusions.]

With this analysis in hand, I now turn to a critique of current methods in robot design, in this section and the two following. In this proposal they are still immature and perhaps poorly explained. The development of these ideas will be a major contribution of my thesis.

4. The Traditional Approach to Robot Perception

In contrast to human perception, most robot visual perception uses a single depth cue and is completely data driven. For example, the most common techniques for recovering depth information are stereo matching and optical flow. They both rely on finding the same object in two or more images, and they do this by template matching between small windows in each image. There are many situations where this doesn't work or is misled. The three most common are areas of low texture (e.g. a flat grey wall); fence post errors, where a feature repeats through the environment (e.g. a fence or row of file cabinets); and the halo effect where one object occludes another.

Perhaps the most explicit statement of this traditional view is still given by David Marr and H. Keith Nishihara [1978, *Visual Information Processing*]. They state "the problem commences with a large, gray-level intensity array, ... and it culminates in a *description* that depends on that array, and on the purpose that the viewer brings to it." The role of expectations and domain knowledge are at best downplayed by this description, and at worst absent. Marr and Nishihara point to constraints such as continuity or rigidness as the basis of a theory of vision, because they are universal constraints, that is, independent of a particular domain or task. Yet, such algorithms work only in limited domains, because those constraints are *not* universal. For example, most scenes contain discontinuities at object boundaries, the ultimate discontinuity. There's little or nothing that's truly universal in perception.

Marr and Nishihara think of vision as the inverse of computer graphics: given an image, and a model of the physical processes that went into producing it, what scene was rendered? However, to be fast and robust in the face of noise and other problems, the human visual system does a lot of "filling in the gaps" (think of the phoneme restoration effect above, or the subjective contours in Figure 5). Our edge detectors aren't simply recovering objective properties of the image or the world, but using the image as one clue, together with a lot of other information. Or consider that the colour of an object (as we perceive it) can change when we put a different colour next to it. To say that people are bad at determining colour or intensity gradients misses the point. Much of the evidence that vision is bottom up comes from experiments where all context is eliminated. Whether it's anesthetized cats starring at line segments or triangles and cir-

So, let's assume the letter detectors return a set of possible interpretations, and our word identification algorithm tries out all combinations of letters, choosing only those that are real English words. One problem with this new theory is that written text isn't always composed of correctly spelled and rendered words from a predefined list. A spelling mistake which substitutes a completely unrelated letter would foil our system, as would new words, let alone The Jabberwocky. Further, this assumes that we can, before any other processing, unambiguously find the division between letters.

In the realm of spoken language, things are even worse. The length of a pause within words is often longer than that between words [Ashcraft 1989, *Human Memory and Cognition*, pp. 390-391]. Our perception of foreign languages is closer to the mark: they sound like a continuous stream of babble. But even if we did somehow start by segmenting the string of sounds into separate words, people don't recognize individual words in isolation. Pollack and Pickett [1964, *Intelligibility of excerpts from fluent speech*] recorded several spontaneous conversations, spliced out single words, then played them to subjects. Presented in this way, subjects identified them correctly only 47% of the time. Success went up as the length of the segment increased.

So how do we recognize words? Consider the following sentences:

I like the joke.
I like to joke.
I like the drive.
I like to drive.

The 'to' or 'the' seems to determine whether the last word is interpreted as a noun or a verb. Indeed, this effect is very strong in English, and can force a word to be interpreted as verb or noun even when it isn't usually interpreted that way:

I like to chair.

However, an experiment by Isenberg *et al.* [1980, *A top-down effect on the identification of function words*] showed that the most common category for the last word can influence whether we hear 'to' or 'the' before it. They presented sentences with sounds half-way between 'to' and 'the', and found that when the sentences ended in words like 'joke', which is more commonly a noun, the ambiguous word is interpreted as 'the', whereas words that are usually verbs (like drive) result in hearing 'to'. So, how we identify words doesn't depend only on physical stimulus, but on their syntactical relations.

In fact, how we identify individual sounds in spoken speech can depend critically on what those words mean. Warren and Warren [1970, *Auditory illusions and confusions*] presented subjects with recordings of sentences. In the recordings, one speech sound was replaced by white noise (a cough works just as well). Here is one set:

It was found that the *eel was on the axle.
It was found that the *eel was on the shoe.
It was found that the *eel was on the orange.
It was found that the *eel was on the table.

In all cases, subjects perceived the "right" sound, the sound that best completed the meaning of the sentence. Apparently, none of the subjects even noticed anything unusual about what they heard.

Even when what's spoken is unambiguous, people use semantics to speed up perception. Meyer and Schvaneveldt [1971, *Facilitation in recognizing pairs of words;* also Meyer *et al.* 1975] gave people two strings of letters and asked them to determine, as quickly as possible, whether or not both strings were English words. (Non-words looked like real words and could be pronounced, e.g. "manty", "cabe"). When both strings were words, reactions were faster and more accurate if the words were related (e.g. nurse-doctor) than if they were unrelated (e.g. bread-doctor).

So we don't identify letters or phonemes in isolation; we identify letters and words together. And we don't identify words (and therefore letters) without also identifying the syntactic and semantic meaning of the sentence. This "conceptually driven" view of perception is the accepted view in psychology. Consider this passage from a sophomore textbook:

> After perceiving the first 150 msec's worth of information in the spoken word, we seem to know what word we are hearing. This certainly is based not only on the phonological cues in the spoken word, but also on the context that the earlier part of the sentence has generated. As you hear "Some thieves stole most of the ...," your syntactic and seman-

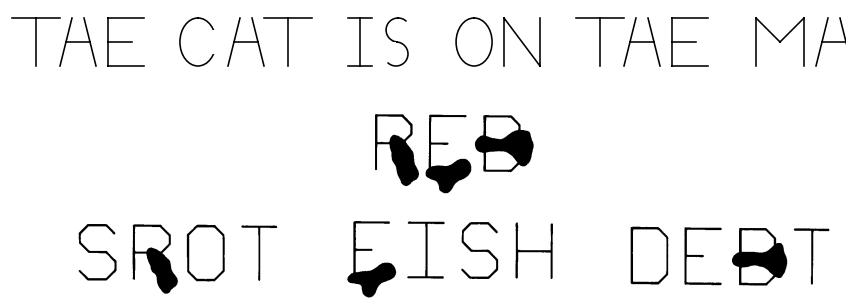


Figure 3: A nieve theory of perception would have it that letters are identified first, then put together to identify words. But as these examples show, word identity can influence letter identity. Note that in the second line, none of the letters are unambiguous, as is demonstrated in the third line.

then two processes are started in parallel. First, the points are stereo matched, which predicts their depth; the focus is then adjusted, and the contrast calculated, for a small number of values around this distance. If the contrast values are unimodal, focusing has verified the stereo match and the result is output; otherwise, the point is rejected. Second, the master camera is focused on each of the original points in turn, to estimate depth. From the depth estimate, the disparity is estimated to predict a window in slave camera and a distance. A number of contrast values in that window are computed for a small number of focus positions around the expected distance. Again, if the values are unimodal the point is kept, otherwise it is thrown out.

Because these experiments were intended to be a first voyage into this technique, it's hard to judge the ultimate potential of the technique. However, a few generalizations can be made. First, the technique involves focusing each of the cameras at many different depths in the scene; this alone can take several seconds. Also, any moving objects in the environment would confuse this technique, since it assumes no motion between the different focus positions. And, as with the techniques above, textureless and featureless areas foil the technique. Finally, the technique hasn't been used in an obstacle avoidance system. As mentioned above, we can't know how well it will work or what its failure modes will be until we close the loop.

While reliable obstacle avoidance from vision may be impossible today, there is a simple proof that it's possible: people do it. Since human perception is still well beyond the state of the art in robotics, we may be able to gain some insight from the workings of human perception.

3. The Nature of Human Perception

Since our goal is perception, let us look at how human perception works. To be specific, let's pick an area in which a lot of research has been done: how people understand the spoken or written word.

Conceptually Driven Perception

An initial, "obvious" theory is that we identify the letters and spaces (for written text) or phonemes and breaks (for spoken text) individually, then put them together to identify words, then figure out the syntax of the sentence, and finally the semantics. This is known as the "data driven" approach. This approach is so self evident that it may seem necessary—how can one identify a word before identifying the individual letters?

There is much evidence, however, that words and letters are identified together. Two famous examples are shown in Figure 3, the first due to Selfridge [1955, *Pattern recognition in modern computers*] and the second from Rumelhart, *et al.* [1986, *Parallel Distributed Processing*], based on Lindsay and Norman [1972, *Human Information Processing*]. In the second example none of the three letters are unambiguous on their own, as is demonstrated by the bottom line. As a third example, we often don't notice spelling mistakes, which evidences the idea that they're corrected early on.

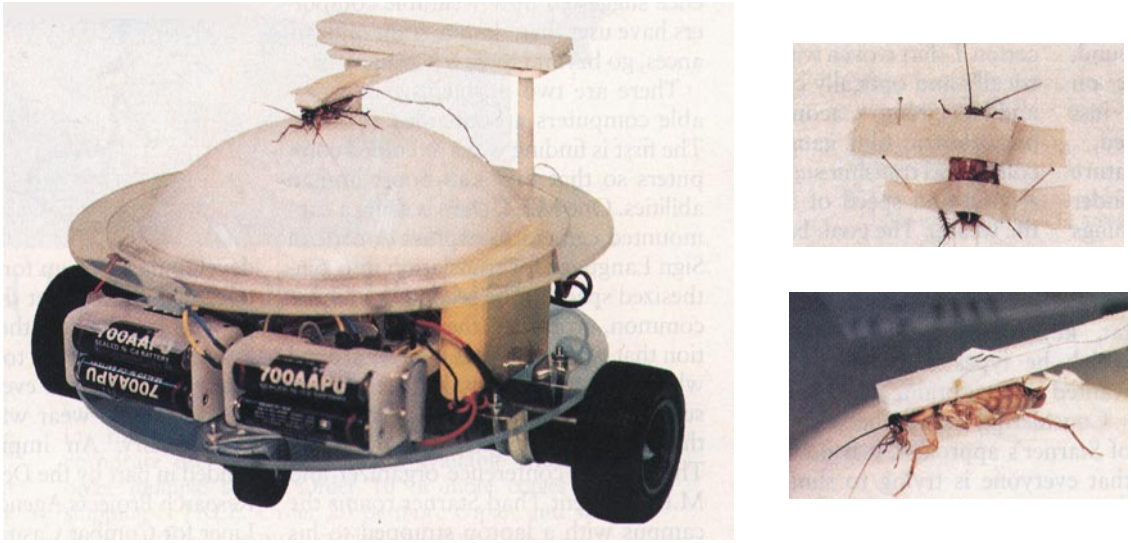


Figure 2: A live cockroach is used to direct this robot base. The roach is allowed to run on the ball, and sensors in its leg muscles are used to steer the base. The sensing, planning and navigation are all done by the roach.

Analysis

As can be seen from the above examples, the problem of navigating from vision alone has yet to be done well enough to leave a robot unsupervised for hours in an unstructured indoor environment. Note that the planetary analog environments tend to be easier than indoor environments, since obstacles tend to be large and have texture, as opposed to objects like tables at eye level or plain walls. As well, planetary environments are static whereas indoor environments are often filled with people.

Looking back on the above examples, it isn't clear that a single depth cue is sufficient for the job. Area based stereo, for example, may simply have too many errors to be reliable for obstacle avoidance. That statement can never be proven, of course, since some day someone may think of a unique twist which makes it work. But from the current state of the art, the sufficiency of stereo is far from certain.

Since ranging devices like sonar (or cockroaches) make the obstacle avoidance problem much easier, it may seem an obvious way to solve the problem. However, there are two reasons to shy away from them. First, they have many problems of they're own. For example, sonar has as specular reflection (many surfaces act as mirrors to ultrasound), poor angular resolution, and an inability to see drop offs. But more importantly, my goal is not simply to engineer a prototype obstacle avoidance system, but rather to figure out how to build a better vision system. It is the new techniques that need to be discovered that interest me, rather than obstacle avoidance itself. I'm interested in creating robots as competent as the cockroach, not using the cockroach without understanding. To put it differently, we do these things not because they are easy, but because they are hard.

As is explained below, one of the novel approaches of this thesis is to integrate several depth cues. Before we leave the previous work, then, let's consider an example of this.

Eric Krotkov with Cooperating Focus, Stereo, and Vergence

Eric Krotkov and Ruzena Bajcsy have investigated combining depth from focus and binocular stereo [Krotkov & Bajcsy 1993, *Active Vision for Reliable Ranging*]. They have a pair of identical cameras on a pan / tilt head. Each camera can be zoomed, apertured and focused independently, and the angle of vergence is also under motor control.

In their experiments, one camera is designated as the "master" camera, the other as the "slave". First, the focus of the master is adjusted to give the maximum contrast. This focal length is converted into a distance, and the slave is aimed at a point that's in the center of the master's view, at the calculated distance. A stereo pair of images is then taken. Points of high contrast are selected in the master image, and

lides with the obstacle. The third cause of failure is the rectangular geometry of the robot base and the presence of bumpers which trigger stopping upon contact. There are instances where the robot is turning to avoid an obstacle, and in so doing, brushes its bumper against a nearby object.

The authors point out that the gaze stabilization, which removes camera rotation with respect to the environment, is crucial for their technique. Even small camera rotations can produce rotational flows that are much larger than translational flows.

Illah Nourbakhsh and Depth from Focus

Illah Nourbakhsh has created a system that uses depth from focus to provide a rough depth map of a scene, and created an obstacle avoidance system based on this [Nourbakhsh 1996, *A Sighted Robot*]. Three cameras are mounted vertically, as close together as possible, and focused at three different distances, namely 0.4m, 1m and 2m. The image is divided into 40 subwindows, 8 across and 5 down. In each subwindow, the sharpness (i.e. how much the intensity changes from pixel to pixel) is computed for the three cameras, with the sharpest giving its focal length to the depth map. The result, then, is an 8 by 5 depth map, where each measurement is "near", "medium" or "far". On flat ground the bottom two rows are medium while the others are far; detecting more than one far in the bottom two rows signals an impending drop off (such as stairs), and the robot turns 180 degrees. Otherwise, the robot turns away from whichever side has the largest number of closes or mediums.

From Nourbakhsh 1996:

> Further indoor tests were conducted in the second and third floor hallways and lounge areas of the Computer Science Department. These tests were the most challenging: lighting conditions and wall texture vary greatly throughout the area. Additional risks included two open staircases and slow-moving students who actively tried to confuse the robot into falling down the stairs.
>
> The robot performed extremely well in this complex indoor domain, avoiding the staircase as well as the students. The robot can reliably navigate from inside a classroom, through the doorway, into the hallway, past the stairs, and into the lounge with perfect collision avoidance in spite of moving students. ... In all three runs [of 20 minutes each], the robot operated fully autonomously and the only environmental modification involved the removal of one coffee table in the lounge that violates our 'beheading' constraint. Average speeds in this domain were approximately 8 inches per second [20 cm/s]. ...
>
> Over several weeks of testing, accumulating more than 15 hours of outdoor time, the robot detected dropoffs and static obstacles with 100% reliability. Furthermore, false positive detection of steps proved to be essentially nonexistent.
>
> Our final experiment involved an outdoor demonstration of the robot... The robot approached the dropoff and the staircase more than fifteen times, detecting them with 100% accuracy. ... Over the course of the demonstration, the robot came in contact with no static obstacles and contacted a moving obstacle (i.e. a human) only once.

Due to its low resolution depth map, the robot has trouble seeing tables near camera height. As well, in areas of low texture, sharpness is low in all three cameras. Such areas are classified as close for safety, but a large patch of such area (such as a plain wall) could paralyze the robot.

Hajime Or at the University of Tokyo

Hajime Or built what he calls a "biomechatronic robot" [Yam 1998, *Roaches at the Wheel*] (see Figure 2) while working on his master's degree at the University of Tokyo. After taping down an American cockroach (*top right*), he inserted fine silver wires into the extensor muscles of the hind legs. The roach was then allowed to run on what amounts to a trackball (*bottom right*). The wires picked up the weak electrical signals generated by the muscles, and the signals were amplified and fed to the motorized wheels. In this way, the machine would mimic the speed and direction the cockroach ran. For the relation of this to the present work, see the next subsection.

agent's environment. In performing the derivation, the designer divides the agent's specialization into a discrete set of reusable transformations, each of which is paired with an environment property that makes it valid. I call such properties "habitat constraints" because the set of such constraints define the agent's habitat.

Note that these are hard constraints: they're built into the very design of the robot. As in the work of Liana Lorigo above, while these constraints are extremely helpful most of the time, the robot's most troublesome failure modes occur where these constraints are violated. No attempt is made to detect possible exceptions, or anything else that might turn these into soft constraints.

Ratler at CMU

The Ratler trials [Krotkov *et al.* 1995, *Stereo perception and dead reckoning for a prototype lunar rover* and Maimone, 1997, *Lunar Rover Navigation 1996*] are part of the Lunar Rover Initiative [Katragadda *et al.* 1996, *Technology Demonstrations for a Lunar Rover Expedition*], aimed at developing technologies for autonomous and teleoperated exploration of the moon. Specifically, the goal is a 200km traverse of the lunar surface that would visit several historical sites. To accomplish this, a stereo algorithm was developed and implemented on the Ratler vehicle, along with an obstacle avoidance planner and arbiter. After rectification, the normalized correlation is used to find the stereo match. The match is rejected if the correlation or the standard deviation is too low, or if the second best correlation is close to the best.

Using this system, Ratler was run at three different planetary analog sites: the Pittsburgh Slag Heaps, the Robotics Engineering Consortium, and the Moon Yard. Travelling at 50 cm / sec over 6.7km the system had 16 failures, for a mean distance between failures of 417m. No information on failure modes is available.

David Coombs and company at NIST

David Coombs' group at NIST has succeeded with runs of up to 20 minutes without collision [Coombs *et al.* 1995, *Real-time obstacle avoidance using central flow divergence and peripheral flow*]. Their system uses optical flow, specifically, divergence of image flow in the direction of the camera's heading, which is inversely related to time-to-contact. Two onboard cameras are mounted one above the other on a single pan motor, and processed separately. The top camera has a 40 field of view, and the lower camera a 115 field of view. The divergence and time-to-contact are computed in three overlapping windows of the central image. In the wide angle image, the process estimates maximum flow in two peripheral visual fields (left and right). Maximum flow and time-to-contact are temporally filtered to reduce errors. Recursive estimation is used to update current flow and time-to-contact estimates and to predict flow and time-to-contact at the next sample time.

Using active gaze control, the cameras are rotational stabilized so that their motion is approximately a translation. If the flow is larger on one peripheral side than the other, then objects in the scene are closer on that side, and the robot steers away. When the camera points too far away from the heading, a saccade is made toward the heading. From Coombs *et al.* 1995:

> Experiments with the obstacle avoidance system were conducted in a laboratory containing office furniture and robot and computing equipment. Furniture and equipment lined the walls and there was free space roughly 5 m by 3 m in the center of the lab. Office chairs provided obstacles. ...
>
> Variability in steering and stopping depends in part on the textures visible from a particular approach. Stopping distances may vary just because the angle of approach varies. As time-to-contact and peripheral flows are computed, the robot moves forward toward open areas while centering itself in the open space. Upon detection of an imminent collision, it turns to avoid the obstacle and continues its wandering behavior. The system has run successfully for up to 20 minutes.
>
> There are three primary factors that lead to system failure. First, there is a variable time delay between detection of an event and the behavioral response to this event. ... Second, the flow computation assumes that there will be sufficient texture in the field of view from which to compute flow. If this is not the case, no flow is detected and the robot col-

Failure modes include objects outside the camera's field of view, especially when turning. The authors suggest that a wider field of view would solve this problem. Other failure modes:

> [C]arpets with broad patterns or boundaries between distinct patterns resulted in false alarms. Sharp shadows also posed a problem in bright outdoor sunlight when shadows were sometimes classified as obstacles. Similarly, bright specularities on a shiny floor occasionally caused the system to falsely report obstacles. Prior knowledge of such patterns or an additional method for depth estimation would be required to resolve these issues.

In this work, an object is anything different from the reference window. This incorporates a kind of domain knowledge, namely that safe ground has approximately the same histogram in all places. However, this is implemented as a "hard constraint": there is no room for exceptions. While this works most of the time, the failure modes above occur when this constraint is violated. One could imagine, for example, trying to recognize exceptional circumstances and developing new rules for them, or having a learning technique that does this automatically.

Ian Horswill and Polly the Robot

Ian Horswill's PhD thesis [Horswill 1994, *Specialization of Perceptual Processes*, and Horswill 1993, *Polly: A Vision-Based Artificial Agent*] provides a vision system similar to Liana Lorigo's. Running on Polly the robot at MIT, the obstacle avoidance software exploited the texturlessness of the floors, and labeled every area whose texture was less than a certain amount as floor. Starting at the bottom of the image and moving up, the first textured pixel in each column was found, and declared to be an object. The closer such pixels were to the bottom of the image, the closer the object was to the robot. By telling what side of the image the object was on (left or right), the robot could veer away from it.

From Horswill 1995:

> In general, all low-level navigation problems are obstacle detection problems. Fortunately, most of these are false positives rather than false negatives so the system is very conservative. The system's major failure mode is braking for shafts of sunlight. If sunlight coming through office doors in into [*sic*] the hallway is sufficiently strong it causes the robot to brake when there is in fact no obstacle. Shadows are less of a problem because they are generally too diffuse to trigger the edge detector.
>
> False negatives can be caused by a number of less common conditions. The present system also has no memory and so cannot brake for an object unless it is actually within the camera's field of view. Some of the objects in the lab have the same surface reflectance as the carpet on which they rest, so they can only be distinguished in color. Since the robot only has a black and white camera, it cannot distinguish these isoluminant edges. The edge detector can also fail in low light levels. Of course, most vision systems are likely to miss an object if they cannot even find its edges so this failure mode should not be surprising.
>
> ... Given the current state of vision technology, it is a bad idea to rely exclusively on vision for obstacle avoidance...

It should be noted, as explained in the sections that follow, that I believe his conclusion about needing colour is suspect. Subjective contours and domain knowledge might do the trick. In fact, if a person could see the leg in the camera image (as displayed on a screen), then it's certainly possible. But certainly in the vision community as it exists today, his conclusions are reasonable.

A main contribution of the work is the idea of specialization. From [Horswill 1994]:

> We can analyze this relationship [between agents and their environments] formally by deriving an agent that is specialized to its environment from a hypothetical general agent through a series of transformations that are justified by particular properties of the

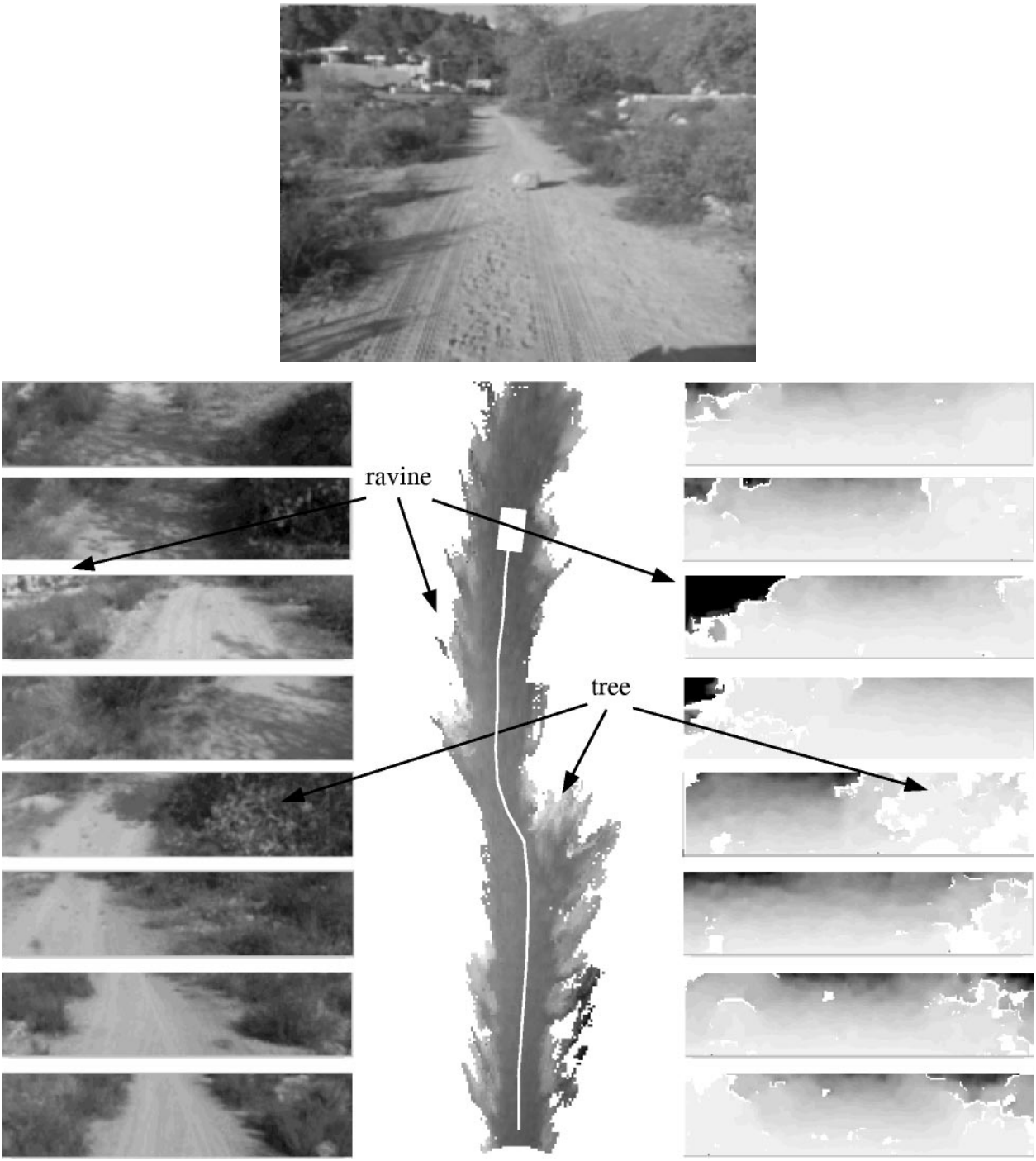


Figure 1: Road run with Ranger, covering about 200 meters. At top is an overview of the road travelled. Shown below are windows of attention from representative intensity images (left side), corresponding range images (right side), and a composite elevation map from the whole run (center). The white curve and rectangle on the elevation map represent the vehicle path and the vehicle itself at the end of the run. From Matthies *et al.* [1995, *Obstacle Detection for Unmanned Ground Vehicles*]

Larry Matthies and company at JPL

This group has built a number of complete systems, all using stereo vision [Matthies *et al*, 1995, *Obstacle Detection for Unmanned Ground Vehicles*]. They first rectify the images, then compute image pyramids, followed by computing "sum of squared differences", filtering out bad matches using the left-right-line-of-sight consistency check, then low pass filter and blob filter the disparity map.

Their algorithm has been tested on a number of robots. The Mars rover test bed vehicle "Robby" accomplished a 100m autonomous run in sandy terrain interspersed with bushes and mounds of dirt [Matthies 1992, *Stereo vision for planetary rovers*]. And the JPL HMMWV drove 200m continuously at 1 to 3 m / sec while detecting half-meter obstacles in time to stop. Figure 1 shows results from a test run that travelled about 200 m along a road. From Matthies *et al.* 1995*:*

> The vehicle was instructed to follow its current heading whenever it could, but to swerve to a new heading as necessary to avoid obstacles. Two significant swerves are seen in this run. The first was to avoid a tree on the right side of the road; the second was to avoid a bush on the left side of the road, just beyond the tree. This was a very successful run, and typical of results obtained in this kind of terrain.

The HMMWV has also accomplished runs of over 2km without need for intervention, in natural off-road areas at Fort Hood in Texas [Matthies, personal communication]. The low pass and blob filtering mean the system can only detect large obstacles; a sapling in winter, for example, might go unseen. Unfortunately, nothing more is said about the performance of the system, or it's failure modes. Sadly, this seems to be the norm in the reporting of such research.

Liana Lorigo at MIT

This Master's and now PhD work under Rodney Brooks and W. E. L. Grimson uses a novel technique [Lorigo 1996, *Visually-guided obstacle avoidance in unstructured environments* and Lorigo *et al.* 1997, *Visually-guided obstacle avoidance in unstructured environments*]. The monocular camera returns 64x64 pixel colour images. The analysis starts by computing a histogram over the 20 (wide) x 10 (high) pixel window in the lower left. The statistic used in the histogram varies; it's either the pair (red / intensity, green / intensity), the pair (hue / intensity, saturation / intensity), or brightness gradient magnitude. This histogram is the "reference histogram" for this column. It is assumed that this area is free of obstacles (or, perhaps, that if an obstacle is that close, nothing can prevent a collision). The window is moved up one pixel and the histogram is recomputed. If it differs from the reference by more than some fixed amount, the new window is said to contain an obstacle and processing stops. Otherwise, the window is moved up one more pixel and the computation repeated.

This gives the location of the obstacle closest to the bottom of the image, for that 20 pixel wide column. The process is repeated for all the columns in the image (that is, starting at horizontal pixels 2, 3, ... 45, not pixels 21, 41, ...). For each column we then have a row number; a flat ground plane assumption is made to translate these into distance to nearest object in that column. Given this, the robot then turns away from the side with the cumulative nearest object.

This system has been tested for more than 200 hours in diverse environments, including test sites at the MIT AI Lab and two simulated Mars sites at JPL. From Lorigo *et al.* 1997:

> For testing in the sandbox, the obstacles were moved into many configurations, additional obstacles were added, and people interacted with Pebbles by stepping in its way. Normally the space between obstacles was only slightly larger than the robot's width. In this situation, the robot navigated safely for large amounts of time. ... The run-time of the robot was limited primarily by hardware concerns and occasionally by the failure modes mentioned below.
>
> Further, the system avoided obstacles in the lounge. Walls, sofas, boxes, chairs, and people were consistently avoided. Corridor following, even at corners, was easily accomplished by the system. Again, moderate lighting variations caused no difficulty. These results were repeated in other rooms where differences included the type and amount of clutter and the pattern of the carpet.

gest problems, other than occasional false matches in stereo or optical flow, are textureless areas and tables at eye level.

Section 3 delves into the nature of human perception from psychology's perspective. The main conclusions are that it's *conceptually driven* and *integrates a number of depth cues*. By conceptually driven is meant that expectations, domain knowledge and task constraints feature prominently in human perception. As well, people use over a dozen different cues to recover relative and absolute 3D information from a 2D image. Although each of them is easy to confuse, together they work rather well.

Sections 4, 5 and 6 form a critique of current methods of creating robot software, as well as some suggestions on how to get around them. In this proposal these ideas are somewhat immature and no doubt poorly explained. I intend the development of these ideas to be a major contribution of my thesis. If validated, this critique is potentially a greater contribution than the prototype system I intend to build.

Section 4 contrasts human perception with current approaches to robot perception. Although researchers agree that integrating depth cues and conceptual information would improve robot perception, they none the less shy away from trying it, due to the complexity. The method that people use to manage complexity is to create *black box* systems, that is, these systems are composed of a number of subsystems that have simple, abstract interfaces. A distinction is made between *hard separation* (found in black box systems) and *soft separation*, where the system is still composed of distinct subsystems, but the systems are more closely integrated. While designing robots by hand leads to hard separation, natural intelligence derives great benefit from soft separation.

Section 5 reviews several existing alternatives in the context of the above critique, namely connectionism and subsumption. Although each addresses certain problems with traditional robotics, they are found to be treating the symptoms, rather than the problem.

Section 6 discusses an obvious alternative to hand designing software architecture: the application of learning techniques to circumvent the above problems. Most learning techniques, however, can't help us. They apply only to a single black box, and expect someone to have already decided on the set of black boxes and their interfaces. By definition, the technique we need must be largely unstructured, since structure is exactly what we want the learning to discover.

Section 7 describes a specific, prototype alternative system that I intend to build for the thesis. Possible inputs to the learning are discussed, as are possible evaluation methods. The proposed system accepts images and the previous depth map as input, and uses them to predict sonar readings, the best direction of travel (as determined by the sonar) and the next image. All of this is to run on the Uranus mobile robot in the Mobile Robot Lab.

Finally, Section 8 lists the exact deliverables and describes a timeline for their delivery.

2. Previous Work

It came as a surprise to me to find that even the basic task of obstacle avoidance from vision alone has never been done reliably: the average distance between failures (i.e. hitting something) is around 400 to 500 meters, for the best systems. At typical speeds of 30 cm/s, this is less than 30 minutes between failures[3]. Perhaps one reason for such disappointing performance is that only a handful of such systems have been built. While there has been much work on creating vision systems, we can't know how well they will work or what the failure modes will be until we close the loop by using them to control a robot. And most systems which do close the loop use some range sensor, typically sonar (for indoor, e.g. Xavier & Amelia) or laser (for outdoor, e.g. Navlabs & Sojourner). In fact, I was only able to find four systems (in the last decade) which actually closed the loop, to which I now turn my attention.

---

3. Average distance between failures is, of course, a horrible metric. A robot that "explores" by travelling in circles could go indefinitely. And in some environments obstacles may be more common, or harder to detect, than others. But the point is that, with the current state of the art, you certainly couldn't trust a vision-only robot in an unstructured environment for even a day.

Breaking Out of the Black Box:
A New Approach to Robot Perception

Thesis Proposal by Martin C. Martin

Draft Four

Surprisingly, the state of the art in avoiding obstacles using only vision--not sonar or laser rangefinders--is roughly half an hour between collisions (at 30 cm/s, in an office environment). After reviewing the design and failure modes of several current systems, I compare psychology's understanding of perception to current computer/robot perception. There are fundamental differences--which lead to fundamental limitations with current computer perception.

The key difference is that robot software is built out of "black boxes", which have very restricted interactions with each other. In contrast, the human perceptual system is much more integrated. The claim is that a robot that performs any significant task, and does it as well as a person, can not be created out of "black boxes." In fact, it would probably be too interconnected to be designed by hand--instead, tools will be needed to create such designs.

To illustrate this idea, I propose to create a visual obstacle avoidence system on the Uranus mobile robot. The system uses a number of visual depth cues at each pixel, as well as depth cues from neighbouring pixels and previous depth estimates. Genetic Programming is used to combine these into a new depth estimate. The system learns by predicting both sonar readings and the next image. The design of the system is described, and design decisions are rationalized.

1. Introduction and Overview

Simply put, my long term goal is to create an intelligent computer.[1] Working on disembodied programs (the traditional domain of AI) usually means working in an abstract world with well defined ontology and action, e.g. chess or theorem proving. When applied to a real world that isn't so simple, the results are often disappointing, e.g. expert systems. As a result, I believe the path to intelligence is through agents embodied in the real world, in other words robots. But robot intelligence is still far behind human intelligence. There are robots that do jobs people can't, but they're usually in hazardous environments, where their material properties are super human, not their smarts: they succeed because of their brawn, not their brains. Even a simple task such as avoiding obstacles using only computer vision is tricky, whether learning is used or not. Solving this in an unstructured world[2] was an early step in the evolution of natural intelligence, and makes a good starting point for robot intelligence.

In the next section, I review the relevant work in vision based obstacle avoidance and find, to my surprise, that the state of the art leads to collisions about every half hour, on average. We're still far from being able to leave the robot unsupervised for hours on end, confident that it won't hit anything. The big-

---

1. The statements in this paragraph are speculative, incomplete and poorly supported, if supported at all. They are meant to describe my underlying philosophical beliefs and provide context for the rest of the paper. If you don't buy them, just accept that it's where I'm coming from and let's agree to disagree. In contrast, the rest of the paper is intended to be full of coherent arguments for various positions. If you don't buy them, let me know so I can adjust my argument, my conclusion or both.

2. In what follows I often talk about "unstructured environments" and "unstructured worlds." For my thesis I take as my prototypical unstructured environment the indoor office environment. It is (arguably) structured by people for people, and contains more geometric shapes than outdoor scenes do. However, it certainly contains some unique problems for visual navigation, including tables at eye level and textureless walls. So, by "unstructured environments" I mean environments that aren't structured for the robot.