Appeared In: *Mobile Robots XIII*, Boston, MA, November 1998, *pp. 126-37*.

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

Martin C. Martin[1]

The Robotics Institute, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 USA

ABSTRACT

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 avoidance 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.

Keywords: perception, vision, fusion, obstacle avoidance, depth cues, psychology, black box, learning, robot, Uranus

1. INTRODUCTION AND PREVIOUS WORK

My interest in this paper is robots which move while avoiding obstacles, their only input coming from video cameras and dead reckoning. In reviewing the literature, it came as a surprise to find that 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.[2] Perhaps one reason demonstrated performance isn't better 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 their 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) or laser (for outdoor). 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.

Larry Matthies' group has built a number of complete systems, all using stereo vision.[1] 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 both a prototype Mars rover and a HMMWV. The Mars rover accomplished a 100m autonomous run in sandy terrain interspersed with bushes and mounds of dirt.[2] The HMMWV has also accomplished runs of over 2 km without need for intervention, in natural off-road areas at Fort Hood in Texas.[3] The low pass and blob filtering mean the system can only detect large obstacles; a sapling in winter, for example, might go unseen.

Liana Lorigo's algorithm[4-5] assumes the bottom of the image represents clear ground, and searches up the image for the first window that has a different histogram than the bottom. This is done independently for each column. If the ground is mostly flat, then the further up the image an object is, the further away it is. The robot heads to the side (left or right) where the objects are higher up.

1. Author Information: email: mm@cmu.edu; Telephone: (412) 268-8713

2. 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.

Figure 1. Word identity can influence letter identity. Note that on the top right, none of the letters are unambiguous, as is demonstrated on the bottom right.

Failure modes include objects outside the camera's field of view, especially when turning. Other failure modes are carpets with broad patterns, boundaries between patterns, sharp shadows, and specularities on shiny floors.

Ian Horswill's algorithm[6-7] is similar to the above. It assumes that the floor is textureless, and labels any area whose texture is below threshold as floor. Then, moving from the bottom of the image up, it finds the first "non-floor" area in each column, turning left or right depending on which side has the most floor.

The system's major failure mode is braking for shafts of sunlight. In addition, it cannot break for objects it has seen previously but doesn't see now. Textureless objects with the same brightness as the floor also cause problems, as does poor illumination.

Ratler[8] used a stereo algorithm to do autonomous navigation in planetary analog terrain. 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. 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' group at NIST has succeeded with runs of up to 20 minutes without collision in an office environment.[9] Their system uses optical flow from both a narrow and a wide angle camera to calculate time-to-impact, and provide feedback that rotationally stabilizes the cameras.

Reasons for failure include the delay between perception and action, textureless surfaces, and hitting objects while turning (even while turning in place).

Illah Nourbakhsh has used depth from focus for robot obstacle avoidance.[10] Three cameras, focused at different distances (near, middle and far), image the same scene. Whichever image is sharpest is the most in focus, so the objects are roughly at that distance. Actually, the images are divided into 40 windows (8 across and 5 down), which are treated independently, giving an 8 by 5 depth map.

In hundreds of hours of tests, the robot has avoided stair cases as well as students, often running down its batteries before a collision. However, failure modes include areas of low texture and tables at the robot's head height.

2. THE NATURE OF HUMAN PERCEPTION

Since our goal is perception, let's look at how human perception works, starting with understanding written and spoken language.

An initial, "obvious" theory of how we perceive language 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.

There is much evidence, however, that words and letters are identified together. Two famous examples are shown in Figure 1, the first due to Selfridge[11], the second from Rumelhart[12]. In fact, how we identify individual sounds in spoken speech can depend critically on what those words mean. Warren[13] 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:

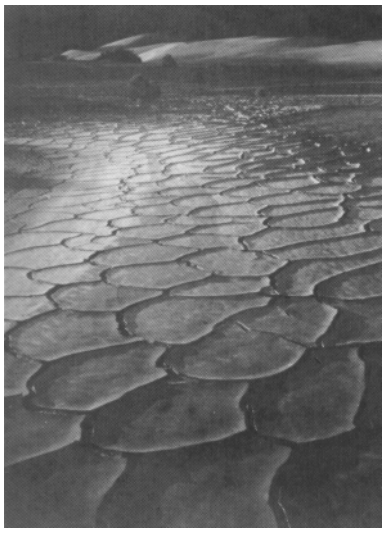
It was found that the *eel was on the axle.

It was found that the *eel was on the shoe.

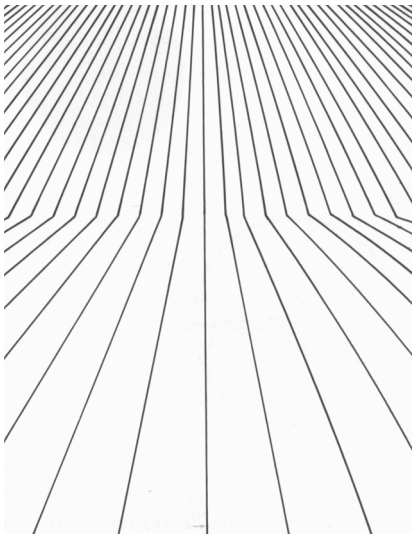
It was found that the *eel was on the orange.

It was found that the *eel was on the table.

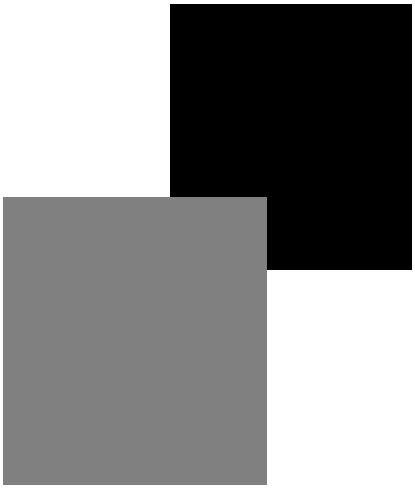
In all cases, subjects perceived the "right" sound, the sound that best completed the meaning of the sentence. Apparently, none of the subjects even noticed that anything was missing.



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

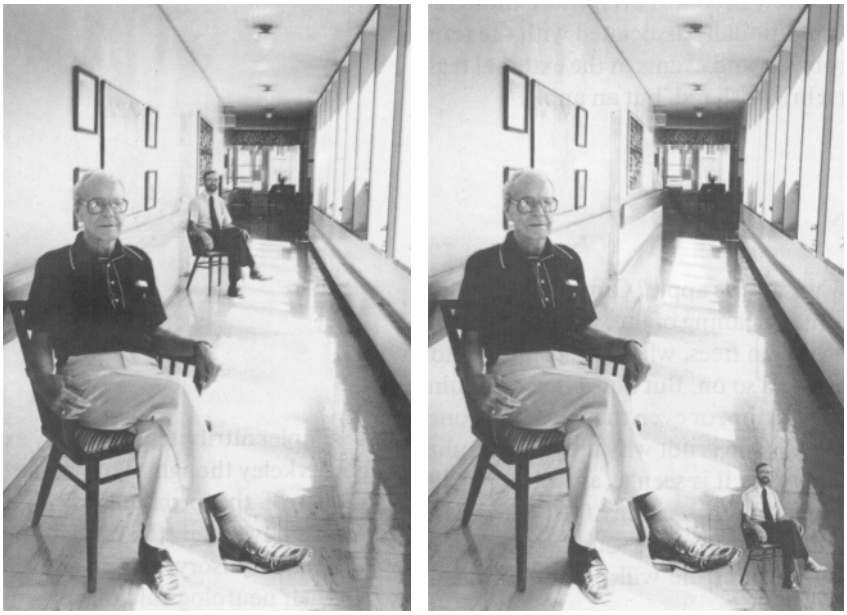


(B) Changes in texture gradients Such changes can be cues to corners, occlusions and other phenomena.



(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 2. An assortment of depth cues.

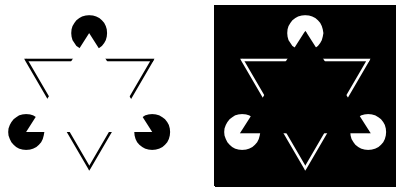


Figure 3. Subjective contours. The three sides of this white triangle on the left (which looks brighter than the white background) are clearly visible, even though they don't exist physically. The effect also works with the intensity reversed.[16]

Even when what's spoken is unambiguous, people use semantics to speed up perception. Meyer[14] 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.

2.1 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 2. 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 toward us. 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.

3. THE TRADITIONAL APPROACH TO ROBOT PERCEPTION

In contrast to human perception, most robot visual perception is completely data driven. For example, the most common technique for recovering depth information is stereo vision. It relies on finding the same object in two or more images, and does 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 when one object occludes another.

Perhaps the most explicit statement of this traditional view is still given by Marr and Nishihara.[15] 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). 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 (see Figure 3). 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 figures of triangles and circles or

anesthetized cats starring at line segments, 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.

3.1 Black box systems

As we've seen above, obstacle avoidance based on bottom up perception doesn't work reliably for long periods of time. Most researchers 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.

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 design 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 need to know much less than 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 motherboards 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 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 `fdiv` statements 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 overall system 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. For software makers to prove that the overall 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.

3.2 The Interaction/Complexity Trade-off

Comparing this with the conceptually driven nature of human perception above, the clash 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 interconnect them, 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, this 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 general, enumerating all the possible interpretations that satisfy the constraint would be far too gigantic a task.

The problem is, the constraint may be easy and natural to state with the concepts of one processing stage (e.g. "It's about the size of a breadbox"), but seem arbitrary or awkward in another stage (e.g."It's a cat or a dog or a footstool or a cardboard box or...).

3.3 Hard Separation vs. Soft Separation

Each black box is responsible for a certain kind of decision (e.g. letter identity, word identity, recovering semantics). Since all the knowledge used to make the decision is embodied in a single black box, its output is typically a single interpretation (or a small set of interpretations). Other boxes accept it as gospel, or reject it only if it results in a blatant contradiction.

One way to provide more interaction 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 non-trivial 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 robot with hard separation; I refer to such interplay as *soft separation.*

3.4 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 about any of the issues of their design. 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. 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. Once in this view, it's hard to see how else the problem could be attacked.

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 they naturally 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. In other words, engineering consists largely of trying to create subsystems that are close to scientific idealizations. Viewing other branches of science and engineering in this light may lead to fruitful insights. However, in this work I'm interested solely in its impact on robotics.

Before exploring some general alternatives, let's examine some existing, non-traditional paradigms in light of this critique.

4. 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.

4.1 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 numbers. It's not clear how to represent plans, for example, in this notation. Also, neural nets can have problems with scaling, unless some *a priori* structure is imposed on the net.

For these and other reasons, a single neural net isn't typically used for a high level task. 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. By this point we already have a black box structure. The neural net is used strictly to implement the black box.

Even so, 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, 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 of 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.

4.2 Subsumption

Subsumption[17] is probably the approach that comes closest to breaking out of the black box. Rodney Brooks' main criticisms of robotics are that representation is overused, and that unlike current robots, the human mind is a collection of a large number of agents. As one alternative he suggests the subsumption architecture.

By allowing higher level tasks to subsume the output of lower levels, they can attempt to correct some of their problems. In fact, each level contains a number of boxes, and higher levels are free to read and subsume any of these outputs. That gives higher levels access to internal details at the lower levels.

However, this interaction can only happen in rigidly specified ways. 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. Since the lower levels are built and tested before the higher levels are designed, there typically aren't a lot of places to inject extra constraints. It's a lot like taking code from one program and trying to use it for a different task: it usually doesn't have the hooks you want, unless it performs a very well specified task (like managing a stack or other abstract data type). And very little in perception is that well specified.

Also, the only way different levels can interact is through reading each other's "internal memos" and replacing them completely. This excludes providing advice or constraints, for example.

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. The input and output semantics of the black boxes are decided by the researcher, and must be clear and simple, because the inputs could just as easily be coming from higher levels.

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" hypothesis can be stated as "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 bottom up vs. 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.).

But this critique doesn't apply to Brooks' criticisms, only to the subsumption architecture itself. The use of representation and the use of black boxes are largely independent. One could easily imagine a collection of agents, that interact in a more interdependent way, and that have little or no central representation.

5. 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 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?

The fundamental difference between nature's and science's approach 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.

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

Learning also does away with the tension between generality and exploiting domain and task constraints. If some 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 can be 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.

This isn't to say that anything goes or that such designs can be arbitrarily complex. As pointed out above, animals have a lot of structure. There are 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, there will be a simple rule that works in most cases. 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 the prototype, 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 learning more palatable, I need to provide some hope that evolutionary computation could produce a practical design in significantly less than a billion years. Here are some ideas.

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, this alone 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. But if 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.

6. A PROTOTYPE

We are currently applying these ideas by constructing a visual obstacle avoidance algorithm for the Uranus mobile robot (see Figure 4). The mascot of the Mobile Robot Lab, it has a 3 degree of freedom base with dead reckoned positioning. While forward/backward motion and turning in place is fairly accurate (about 1% error), sideways motion isn't (10 - 20% error, significant rotation). For sensors it has three monochrome cameras (soon to be color cameras), a Denning sonar ring of 24 sensors, and we plan to add bump sensors to it. Computational resources include a motion controller board, which receives commands from an onboard 68000 based computer running VxWorks. It in turn communicates with a host computer, a dual 333MHz Pentium II running the BeOS.

The robot is largely restricted to the hallway and offices near my office, and of necessity much development and testing will take place in this area. The most problematic visual features are the glossy, textureless grey walls which often confound

local depth estimation techniques such as stereo, optical flow and depth from focus.

Table 1 summarizes the various evaluation methods. The shaded method was selected for the prototype (see Figure 5). In order to assign a competence score to potential programs, our program predicts the sonar readings and the correct direction to travel, and is scored according to accuracy. 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.

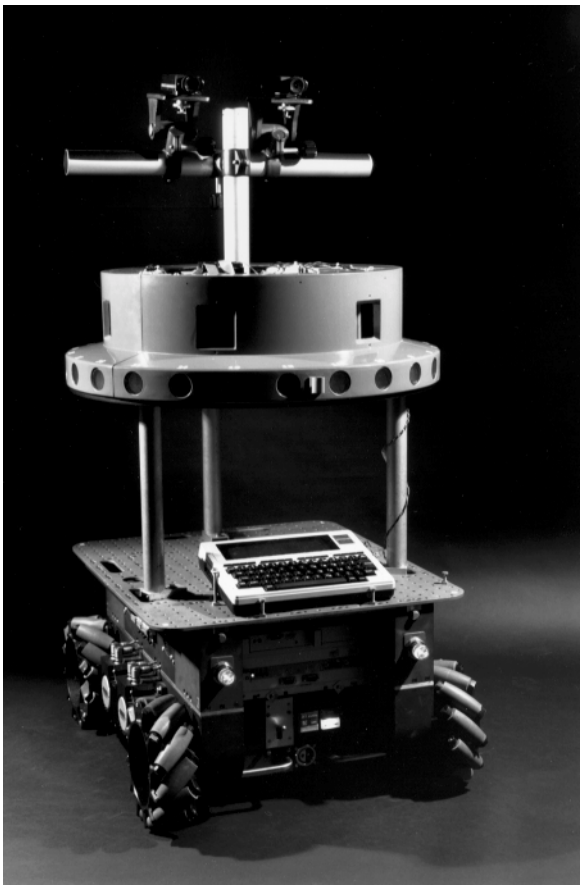


Figure 4. The Uranus mobile robot, showing omnidirectional base and sonar ring. Not shown are cameras, bump sensors and desktop computer.

6.1 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 6), the perceptual system gets the raw image, and for stereo and optical flow, the top 3 stereo matches (with confidences). To reduce the total information, it gets these only every 8 pixels. We also include some measures of texture and gradient magnitude, which are simply convolutions with programmer chosen masks. To allow for expectations, we 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 must 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 walls painted a solid color. 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 hope, however. For example, textureless areas usually represent a smooth change of depth, since physical discontinuities (e.g. occlusion boundaries) tend to produce discontinuities in the image. So, we also include some subset of neighbouring pixel's inputs.

These considerations have lead to the framework of Figure 6. 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 estimates, with 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 work best with 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 only represent discontinuities awkwardly. For example, representing the rule "select the stereo estimate with the highest confidence" (`argmax`) is difficult, 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[18]. Genetic Programming is an offshoot of genetic algorithms which represents a function as its parse tree. Somewhat unexpectedly, Genetic Programming (and evolutionary computation in general) have been shown to be efficient enough for practical use. 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 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.

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/ Hand-made 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

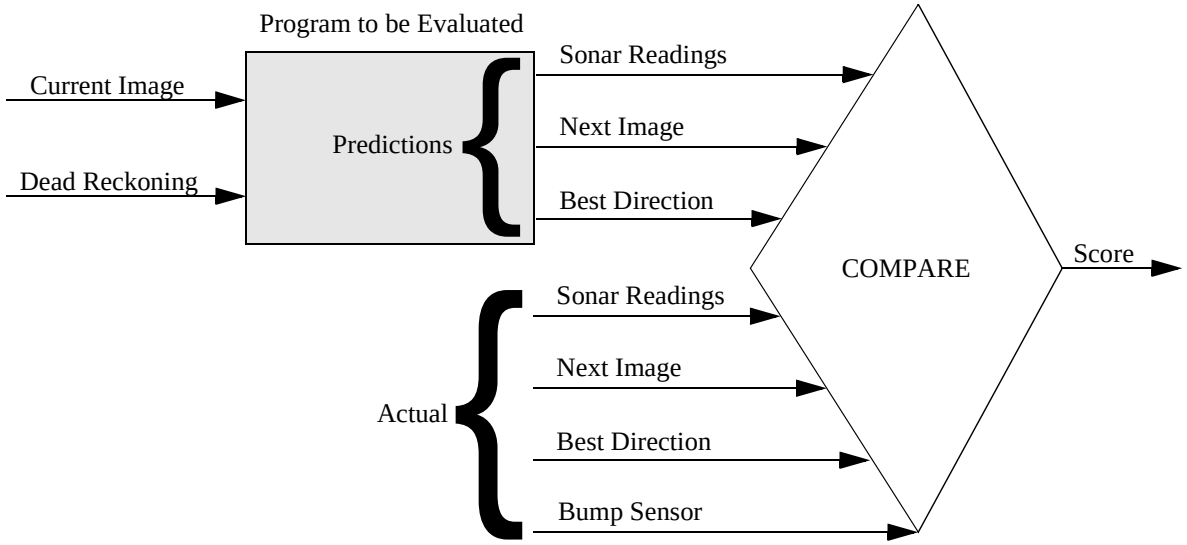


Figure 5. How potential programs will be evaluated. At each timestep, the robot will capture an image; the program will try to predict the current sonar readings, the next image and the best direction to travel. These are compared to the actual values, the robot is moved in the predicted best direction, 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.

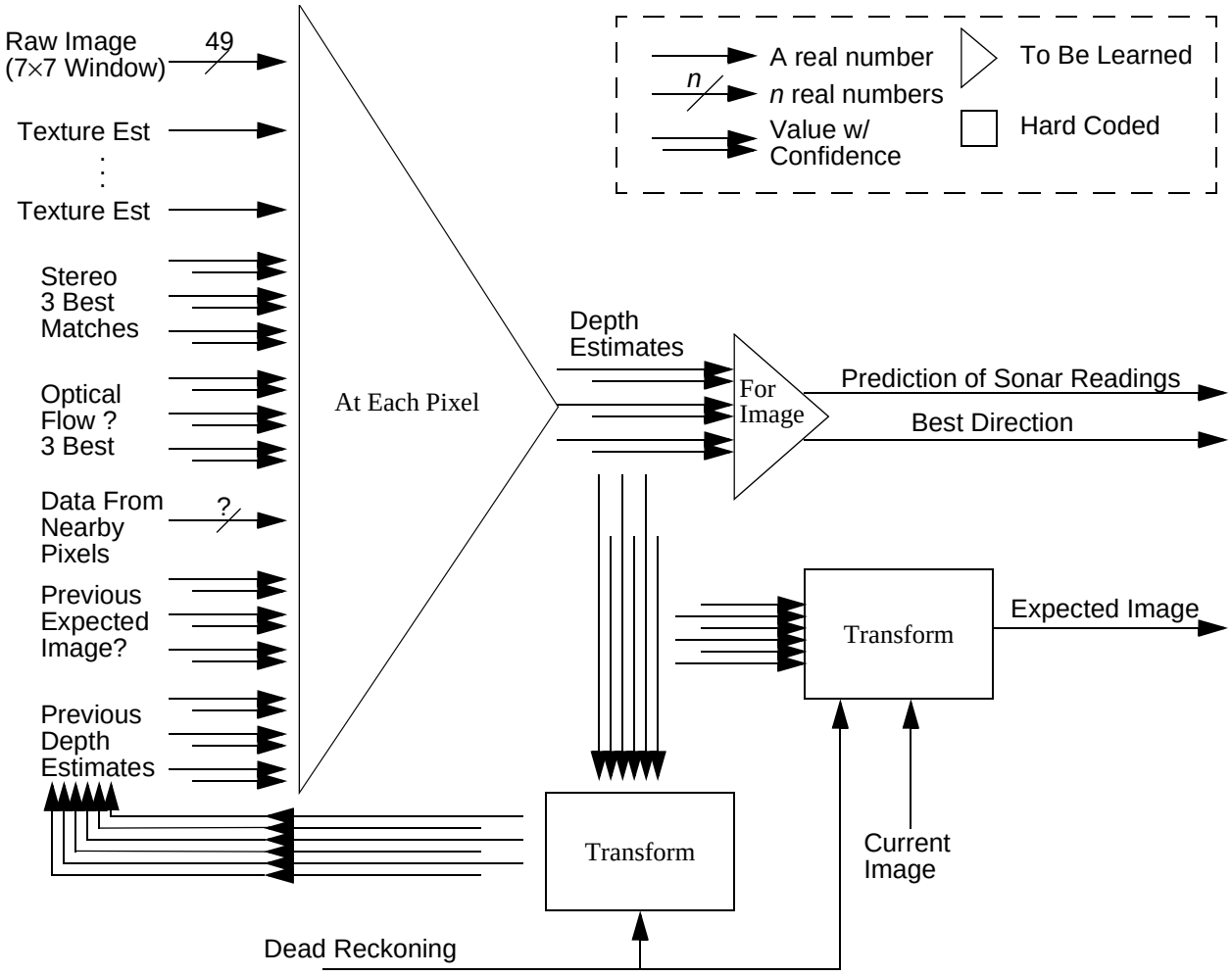


Figure 6. Structure of the perceptual system. The incoming image is processed to compute a number of different depth cues. At each pixel, the same function is evaluated (large triangle). It takes the image, the depth cues, some subset of that from neighbouring pixels, and the previous depth expected at this pixel. 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.

6.2 Summary

Current robots suffer from black box isolation: their various subsystems have very circumscribed interactions. As an alternative, we are currently constructing a prototype system that uses machine learning to perform obstacle avoidance from vision. The system bootstraps itself by predicting sonar data from image data. Specifically, it combines a number of depth cues into a depth map, and then predicts both sonar readings and the next image. From the predicted sonar readings it chooses a direction of travel, and then iterates.

7. REFERENCES

1. Matthies, L. H., Kelly, A., & Litwin, T. *Obstacle Detection for Unmanned Ground Vehicles: A Progress Report.* 1995. http://robotics.jpl.nasa.gov/people/lhm/homepage.html
2. Matthies, L. H. "Stereo vision for planetary rovers: stochastic modeling to near real-time implementation." *International Journal of Computer Vision,* 8(1): 71-91, July 1992.
3. Matthies, L. H. Personal communication.
4. Lorigo, L. M. *Visually-guided obstacle avoidance in unstructured environments.* MIT AI Laboratory Masters Thesis. February 1996. `http://www.ai.mit.edu/people/liana/liana.html`

5. Lorigo, L. M., Brooks, R. A. & Grimson, W. E. L. "Visually-guided obstacle avoidance in unstructured environments." *IEEE Conference on Intelligent Robots and Systems* September 1997. `http://www.ai. mit.edu/people/liana/liana.html`
6. Horswill, I., *Specialization of Perceptual Processes.* Ph.D. Thesis, Massachusetts Institute of Technology, May 1993. `http://www.cs.nwu.edu/~ian`
7. Horswill, I., "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. `http://www.cs.nwu.edu/~ian`
8. Krotkov, E., Hebert, M. & Simmons, R. "Stereo perception and dead reckoning for a prototype lunar rover." *Autonomous Robots* 2(4) Dec 1995, pp. 313-331
9. Coombs, D., Herman, M. Hong, T. & Nashman, M. "Real-time Obstacle Avoidance using Central Flow Divergence and Peripheral Flow." *Fifth International Conference on Computer Vision* June 1995, pp. 276-83.
10. Nourbakhsh, I., "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.
11. Selfridge, "Pattern recognition in modern computers." *Proceedings of the Western Joint Computer Conference.* 1955
12. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Volume 1: Foundations. Cambridge, Massachusetts: The MIT Press. 1986.
13. Warren, Auditory illusions and confusions. *Scientific American* 223, pp. 30-36. 1970.
14. Meyer, D. E. & Schvaneveldt, R. W. Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology, 90,* pp. 227-234, 1971.
15. Marr, D. & Nishihara, H. K. Visual Information Processing: Artificial Intelligence and the Sensorium of Sight. *Technology Review, 81,* 1978, pp. 2-23.
16. Kanizsa, Subjective contours. *Scientific American* 234:48-52. 1976.
17. Brooks, R.A. Intelligence without representation. *Artificial Intelligence* 47 (1991) 139-159. 1991.
18. Koza, *Genetic Programming: On The Programming Of Computers By Means Of Natural Selection.* Cambridge: MIT Press. 1992.