A STUDY OF THE REAL-TIME CONTROL OF A COMPUTER-DRIVEN VEHICLE
SCHMIDT, RODNEY ALBERT, JR.
ProQuest Dissertations and Theses; 1971; ProQuest Dissertations & Theses Global

72-5989

SCHMIDT, Jr., Rodney Albert, 1945-
A STUDY OF THE REAL-TIME CONTROL OF A COMPUTER-DRIVEN VEHICLE.

Stanford University, Ph.D., 1971
Engineering, electrical

University Microfilms, A XEROX Company, Ann Arbor, Michigan

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

A STUDY OF THE REAL-TIME CONTROL OF A COMPUTER-DRIVEN VEHICLE

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

Rodney Albert Schmidt, Jr.

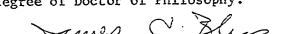
May 1971

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

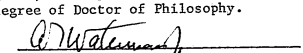
I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

(Principal Adviser, Mechanical Engineering)

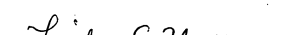
I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Approved for the University Committee on Graduate Studies:



Dean of Graduate Studies

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

PLEASE NOTE:

Some Pages have indistinct print. Filmed as received.

UNIVERSITY MICROFILMS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ACKNOWLEDGEMENT

I wish to thank Professor James Adams for his valuable assistance during the course of this research. His assistance, and that of Professor James Bliss were essential to the presentation of this dissertation in a coherent manner.

My thanks also to Mr. Tony Vacek for his talented reduction of my rough drawings to publishable form in the short time available.

I wish to express my deepest gratitude both to my wife, Marcia, for her support during the years this project was in progress, and to Mrs. Queenette Baur, of the Stanford A.I. Project, without whose dedication and long hours of work this dissertation could not have been completed.

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TABLE OF CONTENTS

Chapter I - Introduction 1
Section A - Background 1
Section B - Organization of this Study 2
Section C - Contributions of this Study and its Relevance to Other Applications 6
Section D - Experimental Framework 8
Section E - Related Research 10

Chapter II - Guidance System 16
Section A - Introduction 16
Section B - Design Criteria for Guidance Using Complex Sensors. 21
Section C - Guidance Algorithm Details 27
Section D - Guidance Error Recovery. 41
Section E - Experimental Performance of the CART Guidance System 44

Chapter III- Navigation 69
Section A - Introduction 49
Section B - Picture Description Techniques and Algorithms 74
Section C - Picture Parsing. 94
Section D - Experimental Results 107
Section E - Suggestions for Automization of the Navigational Decision Process. 129

Chapter IV - Obstacles to Future Work 133
Section A - Dynamic Range of Illumination in Outdoor Scenes. 135
Section B - Stereo and Depth Perception, 3-D Description. 136
Section C - Incident Avoidance, Anticipation, Use of Experience in Decision Making 137

Appendix I - Hardware Details of the CART System . . 144

Appendix II- Discussion of Selected Software Details 151

Bibliography 165

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER I

INTRODUCTION

A. Background

The research described here took place at the Stanford Artificial Intelligence Project during the period 1967-1971. At the instigation of Principal Investigator, Dr. John McCarthy, the author began a study of the computer control of automobiles. The study was initially guided by three major premises:

1. No road modification would be required.
2. Each automobile would be individually automated and would directly perceive all information needed to operate.
3. Each automobile would be capable of completely unmanned operation.

These premises were chosen to guarantee that the study would consider the application of artificial intelligence concepts to vehicle control. Such a control system, if feasible, would have practical advantages, since it would allow computer controlled automobiles to operate in conjunction with conventional automobiles without extensive modifications to the highway systems and without the need for large fixed control networks. The "automatic chauffeur" capability would also allow various traffic control and automobile storage strategies of great interest to urban designers.

As the study proceeded, it became apparent that existing research was not sufficient to allow the definition of such a system. The research

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in this dissertation was therefore undertaken to gain more understanding in three critical and related areas of control: guidance, navigation, and incident avoidance. These three areas are defined in more detail in Section B below. Due to limitations of time and money, the level of investigation varied from area to area. Major emphasis was placed on problems of guidance and navigation.

The work necessarily involved the design and construction of an experimental system including a prototype vehicle. The research was therefore somewhat constrained by the type of equipment available. However, the experimental system allowed the author to validate algorithmic assumptions, ensure that critical problems were not overlooked, quantify hardware and software requirements, and reduce the amount of conjecture which is necessarily a part of an effort such as this.

B. Organization of this Study

Chapters II, III, and IV cover the three major problem areas involved in driving. The division is made by considering the type of information which must be extracted from the environment and the nature of the process required to make use of the information. These areas are:

1) Guidance - the control of the vehicle's motion along a path
2) Navigation - the selection of paths to reach a goal
3) Accident and incident avoidance.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

These functions are required in any system which controls vehicles. In commercial aircraft, guidance is provided in large part by hardware, while the remaining two functions are the pilot's responsibility. In automobiles, all three are provided by the driver. They are explained as follows:

1) Guidance involves the control of the vehicle so that it moves along a specified path. The path is defined by the environment, although artificial aids may be provided to make the guidance easier (I use "artificial aids" to mean things added for no other purpose than to facilitate guidance, such as painted lines). In aerial guidance, artificial aids are almost unavoidable; although the path between New York and San Francisco is defined by the location of the two cities, determining one's position with respect to that path by direct perception of the cities is out of the question. In the guidance of surface vehicles the situation is not so clearcut, and this paper will explore the issue in some depth. The basic similarity between the cases remains this: guidance involves the continuous adjustment of the vehicle's controls to eliminate error in the vehicle's position with respect to the desired path. Except for the problem of error recovery, it is an essentially algorithmic process, involving very little information from the environment.

2) Navigation involves the selection of paths, the recognition of decision points, and the switching of the guidance function from one path to another at discrete times. Here, as before, there are three choices of method: dead reckoning, environmental clues, and artificial aids. Dead reckoning requires very accurate knowledge of path lengths, velocity, time and direction. In general, it cannot be used alone, but must be combined with either of the other methods. Its attractiveness arises from its independence of external sensory input at the time of decision making.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Decision from environmental clues is the natural way to navigate a surface vehicle, since it is the presence in the environment of intersections, driveways and the like which require navigational decisions. Although artificial aids could be provided, existence of the intersections is sufficient, if we can process the information. The attractiveness of using artificial aids versus environmental clues will be discussed in more detail later.

The distinguishing characteristics of navigation are that discrete decisions are required, and that a larger amount of environmental information is required to specify an intersection than to specify a path.

3) Incident avoidance subsumes such things as fixed obstacle detection, trajectory calculation for other moving objects, potential hazard detection and vehicle malfunction detection. Its distinguishing characteristic is that conceptual awareness is required in order to recognize problems and make decisions. This can be seen from the following argument: The thing that "incidents" have in common is that they are unplanned, unexpected situations which require the categorization of an unlimited number of concrete occurrences into action classes based on the nature of the concrete object and the context of the situation. One cannot simply provide a list of all possible concrete situations and instructions for each one. Nor can one provide an abbreviated list and default instructions for situations not covered. First, the percentage of cases which are covered is small, and second, general appropriateness of the default instruction is unlikely.

Not all of these three areas have been explored to the same depth. The work in Chapter II on guidance deals with a conceptual framework

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in which to cast all guidance problems involving complex cues. Conceptually it bears a close family resemblance to feedback control theory, but mathematically the two are virtually incommensurable. Each of the elements of the conceptual framework is analyzed in this chapter and for the most important elements a few examples of earlier systems are shown as they would be portrayed in the new framework. An actual working program is then discussed which carries out the functions described by the framework and operates the experimental vehicle at the A.I. Project.

The work in Chapter III represents an approach to the rough description of complex scenes for use in navigation, allowing for scene motion, perspective, and edge masking effects. The problems of picture processing are sufficiently complex that no conceptual framework is presented here. Rather a particular, although fairly general, method of picture representation and comparison is presented. The chapter discusses the major problems to be overcome in a driving-oriented analyzer, and outlines a particular solution to them. This solution is also represented by a working program, but as yet the program has only dealt with interior scenes. The ideas embodied in the program work, but may prove not to work sufficiently well to drive with the necessary reliability.

The material in Chapter IV differs from the rest in that it does not represent the result of experimentation. With the exception of some work by J. Buchanan at the A.I. Project on obstacle detection by relative motion, no research in incident avoidance has been undertaken. This chapter is of a conjectural nature, detailing philosophical and practical objections to computer based incident avoidance systems. It reflects in part the author's bias against explaining human functions in terms of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

aggregates of neurons in the same way that computer functions can be explained in terms of aggregates of flip-flops.

C. Contributions of this Study and its Relevance to Other Applications

There are several advances contained in this research which stand by themselves and which have applicability beyond automobile control. First is the demonstration that picture processing of relatively complex scenes can take place in real-time. Prior to this study, picture analysis techniques were not used for applications requiring rapid results. As a result of this study, new areas of computer automation can be undertaken with the knowledge that computer processing is a viable method of real-time control.

Secondly, the framework for complex guidance systems presented in Chapter II draws more closely together the techniques of feedback system analysis and programs for computer automation. Formerly, computer control programs were either trivially simple (conceptually), or completely ad-hoc. With no formalism behind their structure, the more complex control programs did not lend themselve to extension, modification or even comprehension. My control system schema will allow control systems to be designed and discussed with the same flexibility now enjoyed by compiler writers as a result of the formalization of compiler structure and computer languages.

Thirdly, the development of region analysis and approximate description of pictures given in Chapter III shows the potential usefulness of picture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

analysis in cases where complex visual data must be processed. This work shows that region analysis permits the rapid extraction of major picture features without requiring foreknowledge of geometric properties of the input scene. This chapter also shows that the time required for region-oriented analysis is approximately equivalent to the time for edge-oriented analysis, making them equally feasible for real-time applications.

The last area of contribution is the relevance of the study as a whole to areas other than computer-driven automobiles. There are at least two other applications in which reliance on external control aids is undesirable, and complex decision problems need not be automated. These are the fields of remote exploration and industrial control.

In the field of remote exploration, the complete control system must obviously be contained within the mission package. The communications propagation delay involved in a Mars mission (up to eight minutes) makes direct Earth control impractical. However, excessive caution on the part of the computer control system creates no hazard, and high level decisions about hazards can be relayed from Earth. The infrequent occurrence of such high level problems prevents the time delay from significantly degrading performance. The work of this study could be readily extended to the automatic selection of a path toward a visible goal, and the maintenance of a course with respect to a visible object.

In the field of industrial control, this study shows that one could consider such applications as automatic warehousing or mail delivery without the necessity of modifying the permanent structure of the building to incorporate guidance aids. Since the environment is restricted and the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

employees aware of the nature of the vehicle, hazards can be minimized in ways not possible in the automatic automobile's environment.

D. Experimental Framework

Due to the prior experience of the A.I. Project with computer visual perception via TV, and Dr. McCarthy's and my belief that an imaging system of perception would be essential to the task, an experimental vehicle was equipped with a TV camera from which most, and if possible all, of the computer's input data would be obtained. The vehicle itself was obtained from the Stanford Department of Mechanical Engineering. It had been previously used for a study of remote control problems produced by the transmission time delay encountered by a vehicle on the Moon controlled by an Earth operator. My thesis advisor, Dr. James Adams, of the Stanford Mechanical Engineering Department, was one of the participants in that study, and was interested in the car study (hereafter called the CART project) because of its obvious applications to remote explorations of the planets, where the time delays, varying from a few minutes to hours, make direct Earth control impossible. My own interest was engendered by the possible applications of this work to industrial processes, where up until now automation has been confined to more-or-less repetitive or mathematically simple operations.

After modification for computer control, the speed of the CART was about 5 mph or 7 ft/sec., thus establishing a maximum processing time of a few seconds for any real-time operation. Since the processing was to be done on the time-shared PDP 10/6 system at the A.I. Project, the control program would get an unspecified fraction of the available processor cycles. Thus the programs had to be carefully structured to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

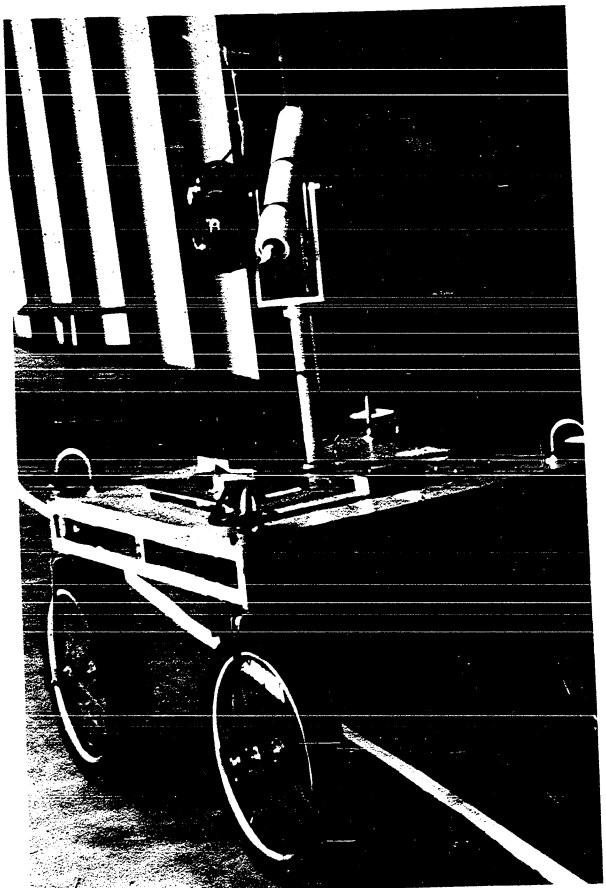


Plate 1 - Experimental Vehicle

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

make important decisions first, to keep the CART from being damaged in the event of difficulty, while at the same time avoiding unnecessary stops due to overcautiousness. The maximum program size is 76,000 36-bit words and the instruction time about 3 microseconds. Use of the maximum program size results in unacceptable service from the time-sharing system, so minimization of program size was another goal. Because of the need for rapid execution, the control programs were hand-coded in FAIL, the assembly language for the PDP-10.

E. Related Research

The research reported here is to the author's knowledge unique, in that it deals with a process (driving) which must be carried on continuously for long periods with a very low error rate. It involves a method (picture analysis) which is ordinarily associated with processes (such as block-stacking) in which the analysis can take place prior to the initiation of action, with no particular constraint on the time available for analysis.

The work is set in the context of a great deal of research in picture processing and computer graphics, work in automatic vehicle control (primarily of a non-computer variety) and a few studies in computer "robotics" centered primarily at MIT's Project MAC, SRI, and at the Stanford A.I. Project. The CART project did not draw upon the sources to be cited here, since in most cases the application constraints were quite different, but an occasional thread of a common idea can be seen.

In the realm of picture analysis and description, a case in point is the work of Zahn [1]. Zahn covers the problem of reducing a picture to a structured description without loss of information. He correctly

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

recognizes that objects may be described by invariant parameters - he calls them "signatures" - and notes the problems caused by aberrations in the input data which change the basic structure of the picture. However, his description does not consist of signatures; rather he calculates signatures from the description. The description itself contains all the information in the original picture. Further, he only considers 2-valued pictures, and does not propose solutions to the problem of structural aberrations.

In a natural picture, not only are there more than 2 gray levels in the input data, but object boundaries are irregular, containing large amounts of "information" which is worthless, because it is non-repeatable. The picture description used in the CART project explicitly throws away a great deal of "information" about the fine structure of objects. This has the dual benefit of making it possible to store a great many pictures in a reasonable amount of space (about 200 words for a 200 x 200 picture), and also attacking the problem of structural aberrations by selection "signatures" less susceptible to change and comparing structures of signatures in a way that allows for many such changes.

Another commonly-quoted work in the field is that of Shaw [2]. This work deals with descriptions of graphs which may be interpreted as pictures. Basically it is a discussion of an application of conceptual representation (as modelled by BNF grammar) to graph representation. It is not applicable to natural pictures, and is also an "after-the-fact" description more suited to the generation of a picture from a description than the other way around. By this I mean that the choice of non-terminal symbols depends on a prior knowledge of the scene - a situation which does

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

not obtain in driving.

Both of the preceding works concentrate on orthogonal projections of 2-D scenes. In the CART work, I treat a scene as a perspectively transformed 2-D scene - a close approximation for the near field of a road scene. Under these conditions, a graphical description of a scene is of no use, since objects change size with distance, and so while the thinking behind the work of Shaw and Zahn is relevant, the actual techniques are not. Although some work in 3-D pattern recognition in ongoing at MIT and Stanford, I do not discuss it here, because I have no practical way to get depth information, and not enough time to process it if I did.

In the field of "robotics", or more properly, computer-controlled manipulators and vehicles, work at MIT, SRI and the A.I. Project should be cited.

Both MIT and Stanford have computer-controlled arms capable of simple manipulative tasks. The original work at Stanford was done in 1967 by W.M. Wichman [3]. Various elaborations have occurred since then, but the basic scheme is the same. Using the known geometric properties of cubical blocks, and the known relationship of the TV camera to the scene, Wichman is able to calculate from an input TV picture the motions of an electric arm required to stack one block on top of another. The computation takes place in advance of the actual stacking, with the arm out of the picture. No attempt is made to use the camera to gain information about inaccuracies in the arm, although repeated tries to improve the stack are made if required. It should be noted here that the "feedback" referred to in the title of Wichman's thesis does not

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

refer to the motion of the arm, but to the determination of the desired arm position. The arm is fitted with potentiometers on the joints, and is servoed conventionally. The major assumptions of this work are simple geometric scenes, and plenty of time to compute very precise solutions - none of which apply to a driving situation.

The work at MIT is similar, with the exception of a program which moves a bucket in one dimension to catch a thrown ball. Here the problem is solved in real time, but the input scene is trivially simple. Reports on this, available to me only by personal conversations in the A.I. "grapevine" indicate a somewhat intimate relationship between the probability of a successful catch and the ratio of bucket diameter to the total length to be guarded!

Besides the work at the Stanford A.I. Project, the only other significant work in computer driven vehicles is conducted at SRI. Their vehicle, equipped with a TV camera and an electronic rangefinder, is described by Nilsson [4] in a paper submitted to the International Joint Conference on Artificial Intelligence. At SRI, information is gathered sparingly from the TV camera and rangefinder (as well as "feelers" connected to microswitches) and used to build an internal model of the restricted experimental space. Once this occurs, TV data is only infrequently required, with most problems being solved by reference to stored data. Objects in the work space are again simple geometric shapes, with the research emphasis placed on problem-solving rather than perception. The work bears a closer resemblance to block-stacking than to CART research, since there is no real time constraint, and geometric objects are used.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

One last piece of work in robotics should be mentioned, primarily because of its wide circulation - it was reprinted in ANALOG SCIENCE FICTION Magazine. This is the work of Sutro and Kilmer [5] relating to the reproduction of human neurological capabilities with computers. This work, among others, is sometimes used to argue that really computers can be organized just like people and thus it is both economical and practical to attempt to mimic any human function with a computer. Except for its emotional appeal to researchers this viewpoint contributes very little to the methods of solution of control problems by computer.

The third research domain relevant to CART research is the automated highway studies conducted over the past several years by many groups. Only a representative study is cited here. Fenton, et al, [6] at Ohio State implemented a system of lane guidance and speed control using a buried cable for lateral guidance and a cord stretched between cars for speed control (to be replaced by a rangefinder in a real system). Fenton's own assumptions make his approach unequal to the task Dr. McCarthy set - Fenton proposes automating only limited access highways, and using human drivers on other roads. As an industrial acquaintance of mine said "If you gotta pay a guy to sit there, he might as well do something useful." This economic objection to such a partial system is a devastating one, to which no satisfactory reply seems possible.

The Ohio State report also suggests that the computers for the complex decision functions be attached to the highway system rather than to the vehicle. This negates their own point about the need for gradual introduction and changeover. If an enormous investment in roads is required before an automated vehicle becomes usable, the economic incentive

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for proceeding is much reduced. The practical problems involved in surveillance of an entire road for dropped objects and intruders, as well as the detection of erratic drivers of uncontrolled vehicles also weigh heavily against this sort of arrangement. At best, the Ohio solution increases the permissible traffic density on superhighways without affecting urban traffic congestion, and at worst it encourages drivers to spend even less time thinking about their driving, with potentially harmful effects on the accident rate. The functions described in this study are not sufficient to constitute a total system, and if a computerized total system is developed, the implementation used at Ohio State is unnecessarily complex. The CART study shows that a digital computer system performing the same functions in the framework of a complete system would require no special road preparation and only a small fraction of total computer capability.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER II

GUIDANCE SYSTEM

A. Introduction

Guidance is the process of making small corrections in a vehicle's controls in order to keep the vehicle moving along a desired path. In the case of an automobile, the path is specified in some way external to the guidance system, and this chapter deals only with this type system. The feedback control system diagram of Fig. 2.1 represents such a system.

Unfortunately, the level of detail of Fig. 2.1 is inadequate to characterize guidance systems of the type described in this chapter. In order for the diagram of Fig. 2.1 to be useful in explaining a system, the processes represented by each of the blocks in the diagram must be simple algorithmic transformations. In the case of recognition of road edges in high noise level environments, the transformations from the actual road edge location to an internal representation used by the control system may be quite complex. The transformation may involve alternate strategies based upon adaptive parameters, sequences of heuristic approximations, or selective use of input data based upon prior experience. The formalism of Fig. 2.1 is not sufficient to discuss such areas.

Fig. 2.2 is an alternate scheme of describing control systems which allows for complexities in the sensor and control transformations and provides a convenient framework from which to discuss them. Section B of this chapter is a discussion of the elements of this framework as they pertain to general vehicle control, while Section C focusses upon the particular vehicle used in this study. Fig. 2.2 tacitly assumes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

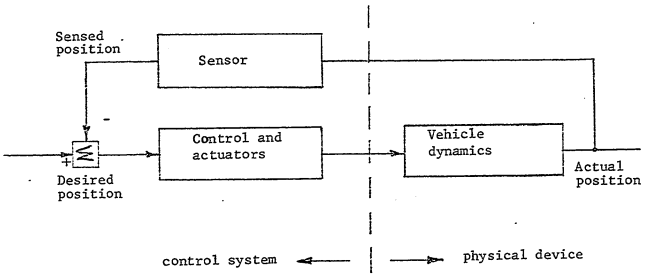


Fig. 2.1 - Feedback Control System

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

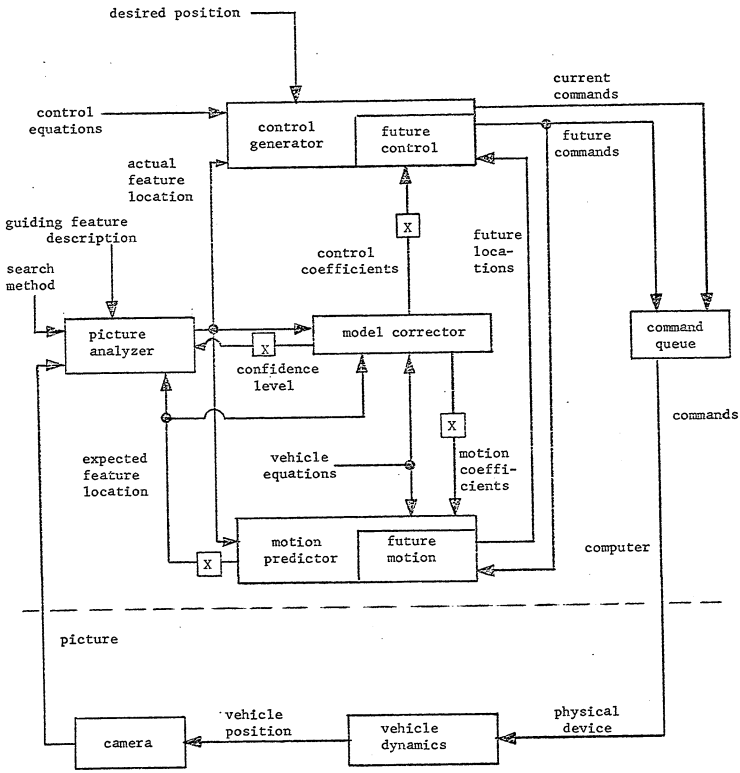


Fig. 2.2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

that guidance is a continuously maintained process. Since all processes must start, and many must restart after errors, Section D discusses error recovery procedures and ways to buy extra time for error recovery processing to take place. Section E is a collection of simulated vehicle runs illustrating various features of the actual CART vehicle control program, which evolved from the considerations of Sections A, B, C, and D.

A.1 System Theory and Guidance with Complex Sensors

By applying the mathematics of system functions to the diagram of Fig. 2.1 one can calculate the dynamic response of the system it represents. If the sensor is simple enough, its output is some mathematically reasonable function of its input, and the system function can be easily calculated. If the input is complex, as it is when using a TV camera to guide a road vehicle, the mathematics of the sensor and accompanying analysis program is no longer reasonable. For the purpose of determining ideal system response, the sensor complexities can be assumed away, but the resulting theoretical vehicle motions bear little resemblance to those actually observed. In a complex problem such as computer vehicle control, the approach of Fig. 2.1 is of little value, since most of the common assumptions underlying feedback system design are invalid. Typically, the forward path system function is assumed to be only approximately known but nonlinear. Normally, the forward path is not considered to be time-varying, thus allowing any serious offsets to be compensated for once and for all. The feedback path is almost always some simple passive system with low noise and excellent linearity. The whole point of the feedback approach is to gain for the entire system the noise reduction and improved

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

linearity made possible by the dominance of the feedback element in the overall system function.

In the present study, none of these assumptions are valid. In the forward path, the system function is affected by many factors outside the control of the system designer. Changes in vehicle loading, wind, road surface composition, road tilt, and aging of the vehicle and its actuators will all introduce variations in the forward path transmission function. An unwary designer might try to swamp out these factors by the use of large loop gain. However, the random error in the sensor and associated algorithms is appreciable, and large loop gains would couple this error into the system with large magnitude. For this reason, the loop gain must be kept low, and the errors in the forward path compensated for by adaptation of control parameters. The approach of Fig. 2.2 makes this process explicit.

In Section C, along with the system developed for the Stanford experimental vehicle, a hypothetical system using photocells is discussed. This is done purely for the purpose of illustrating the scheme of Fig. 2.2. My schema is not intended to deal with systems for which the methods of Fig. 2.1 are adequate. My schema applies to control problems in which

1) the characteristics of the dynamic elements of the system are not measureable or are time-varying, or
2) the sensor reliability is not good, or
3) a requirement for continuity of operation does not permit "retuning" of the system to cope with varying conditions.

Fig. 2.2 does not imply a necessary commitment to computer control,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

although this study deals only with computer control. As a practical matter though, I would expect that most systems complicated enough to benefit from this approach would be computer systems.

With the purpose of the scheme of Fig. 2.2 outlined, let us proceed to a discussion of the roles of the various elements of that figure. In the next section I will treat the elements in a way appropriate to generalized vehicle control systems. This treatment could be generalized still further to deal with other types of control problems, but in the interests of explanatory clarity I have not done so. I leave for Section C the application of Fig. 2.2 to the Stanford experimental vehicle.

B. Design Criteria for Guidance Using Complex Sensors

B1. The Picture Analyzer

The purpose of the picture analyzer in Fig. 2.2 is to find the guiding feature in the input picture. Four pieces of information completely describe the picture analysis system.*

1) the nature of the feature to be found

2) the search algorithm used

3) the expected location of the feature

4) the confidence level of the expected location.

One must choose which portions of this information to "build in" to the system, which to let the operator select, and which to make program-modifiable. Clearly, making these items program-modifiable results

*Of course, one could replace 3) and 4) with a probability map of the entire picture showing the probability $p(x,y)$ that the guiding feature was at location (x,y), but it would take as long to search the map as to search thepicture so there wouldn't be much point. Fortunately, $p(x,y)$ is single humped, so 3) and 4) really contain all the important information.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in the most flexible system. However, the high level processing required to make appropriate modifications is time-consuming and may in fact be impossible. At the other extreme, a simple photocell tracking system, such as the one described in Section C, has all this information built in at construction time. However, the resulting system is so inflexible and lacking in error recovery that it is almost useless, except in trivial applications. A more reasonable method is to "build in" a selection of algorithms which identify the types of features required by the particular problem and have the operator choose the correct algorithm for each particular application. Alternatively, the operator can "point out" the guiding feature and have the computer cycle through the algorithms to find the one which works best on that feature. Either of these methods are equivalent to selecting items 1) and 2) in advance, since the algorithms are written in advance and only work for specific types of guiding features. If during operation something new comes up, the system will not be able to use it for guidance. These methods are attractive when the desired guiding features have some nice mathematical description which results in fast algorithms.

If the features have no nice properties (for example, suppose the feature is an irregular hole in the ground, and one wishes to circle it at constant radius) then an alternate technique is to "build in" some general search algorithm (such as 2-D correlation) and feed it a mask corresponding to the feature involved. The masks can either be provided by the operator, or a higher level program can generate them by examining the guidance feature selected by the operator. It should be pointed out that the use of correlation techniques in a real-time control makes a heavy demand on the prediction portion of the system, since the time

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

involved in a large computer correlation is unreasonable. Thus the predictor's confidence level must be high enough to restrict the search space to a relatively small portion of the picture.

In all cases, items 3) and 4) are calculated by the system, since they change dramatically during operation, and it would make no sense to try and fix them beforehand.

B2. The Motion Predictor

The function of the motion predictor is to use information about the real system dynamics to predict where the guiding feature will be in the input picture when the picture analyzer is next activated. This enables the use of a wide-field sensor without complications introduced by spurious objects resembling the guiding feature. It also permits more exhaustive picture analysis by restricting the area over which the analysis must be applied. It should be mentioned here that a wide-field imaging sensor such as a TV camera with a wide-angle lens is virtually indispensible for higher-level processing such as error-recovery, and is certainly desirable even for tracking (since it permits the guiding feature to move large distances in the field of view without requiring mechanical tracking).

The prediction is determined by four items:

1) the structure of the vehicle's motion equations
2) the coefficients of the motion equations
3) the current position of the vehicle
4) the control inputs to the vehicle.

The structure of the equations is a function of the geometry of the vehicle and cannot in general be deduced from an analysis of the actual

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

motion. Since the system is designed for a particular vehicle, the most reasonable approach is to "build in" the structure of the equations.

The current position and control inputs are of course calculated by the program, since they are the primary input and output of the whole guidance system.

A further use of the motion predictor is to improve system performance by predicting into the extended future where the guiding feature will be. Then if the picture analyzer fails to find the guiding feature, the predictions can be used to keep the vehicle in motion while higher level processes attempt to relocate the guiding feature. This kind of low-level error recovery is essential to provide reliable driving performance. A delicate balance must be struck here to prevent the actual disappearance of the guiding feature from being ignored too long, while at the same time avoiding frequent halts due to temporary interruptions in the guiding feature (for example, breaks in a line on a road).

B3. The Control Generator

The control generator corresponds most closely to the conventional feedback control system. It constructs an "error signal" which is fed to a set of control equations of the form

$$C = \alpha E + \beta(\partial E/\partial t).$$

Where C is the steering wheel angle, E is the lateral displacement from the desired position, and α and β are the displacement and displacement rate control coefficients. In the actual control, $\partial E/\partial t$ is replaced by the angle between the guiding feature and the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

longitudinal axis of the vehicle. This is an equivalent, but much less noisy, rate measure than differences of successive values of ε. The four relevant pieces of information are, of course:

1) the control equations
2) the control coefficients
3) a description of the desired spatial relationship of the guiding feature and the vehicle
4) the actual relationship of the guiding feature to the vehicle.

In most controllers, all of these except the actual relationship are built in, with possibly the desired relationship adjustable by the operator over a small linear range. In a general-purpose programmed controller, not only the value of the desired relationship (the linear spacing from a line, for example), but the nature of the relationship would be variable (i.e. changing from circling a point at constant radius to remaining equidistant from two markers). At the very least, the coefficients of the control equations must be program-variable if the vehicle is to function in spite of variations of a factor of 100 in speed, a factor of 10 in traction, and a factor of 10 or more in maneuvering room available.* The kinds of tracking response needed under these widely varying conditions would be difficult to obtain with a fixed system.

---

* Freeway speed: 70 mph
Parking speed: 0.7 mph
Coefficient of friction
of dry road 1.0 (approx.)
of icy road 0.1 (approx.)
Maneuvering room available
on 10 ft. freeway lane 2 ft
in parking space 3 inches

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In conjunction with the motion predictor, the control generator can be used to generate lists of hypothetical commands to be used in the event of difficulties or delays in picture analysis.

B4. The Model Corrector

The function of the model corrector is to change the internal parameters of the program so that the computer's control and predictions correspond more closely to the behavior of the physical system, and compensate for any errors in the hardware. This is at best an inexact process, since the number of imprecisely known physical constants is larger than the number of reasonably noise-free measurements that can be made. For example, in our robot vehicle, there were originally seven imprecisely known variables (the offset and proportionality constant for each of three controls plus the vehicle speed) and only four measureable variables (the value and first derivative of the slope and intercept of the line being followed). Two sorts of approximations can be made to improve the situation. The first is to use one's knowledge of the physical system to select the parameters most subject to change and adapt only those. With this method, the other parameters must be carefully calculated beforehand. The other method is to divide the responsibility for measured error among the possible sources according to some fixed scheme, and adapt all the parameters. If the parameters are recursively low-pass filtered (as they must anyway to reduce the effect of random errors in measurement), the immediate effect of adaptation errors will be small and the system may eventually converge.

On the other hand, the system may become trapped in various local optima with relatively gross errors in certain parameters. An example

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of this might be the following of a left-curving line. If the prediction is made assuming that the line is straight, the control equations will adapt to allow for a right offset in the wheels. Then when a sharp right turn is called for, the left offset in the model will prevent a sufficiently sharp turn and disaster will result.

For this reason, certain key parameters may have to be left unadapted, even if they are wrong. At the very least, stiff bounds must be placed on the amount of adaptation allowed. Additional reliability can be obtained by designing the physical system with fewer sources of error. For example, the A.I. Project CART was redesigned with only three imprecisely known constants, which improved performance greatly.

I have discussed the main elements of the system shown in Fig. 2.2 from the point of view of overall design, flexibility and purpose. Let me now re-examine these same elements as they were specifically applied in the CART project. In this connection, certain hardware systems will be discussed in terms of the equivalent programmed systems in order to bring out their fundamental nature.

C. Guidance Algorithm Details

C1. Picture Analyzer

A simple 2-photocell tracker, such as shown in Fig. 2.3 has the voltage-displacement relation shown in Fig. 2.4 when illuminated by a point source of light (a light "impulse function"). The output function when tracking a line appears as shown in Fig. 2.5.

The distance "L" is the line width, "W" is the photocell width, and x is the distance from the center of the line to the center of the photocell pair. Such a sensor would ordinarily be used to keep the line

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

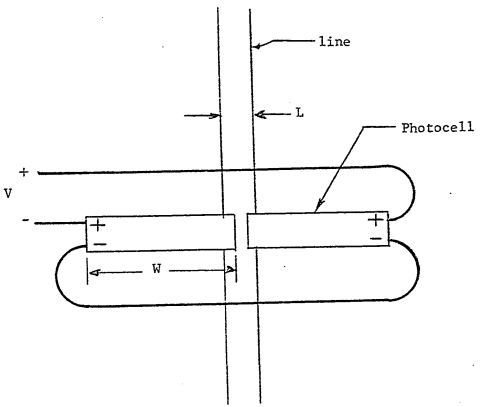


Fig. 2.3 - Simple Photocell Tracker

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

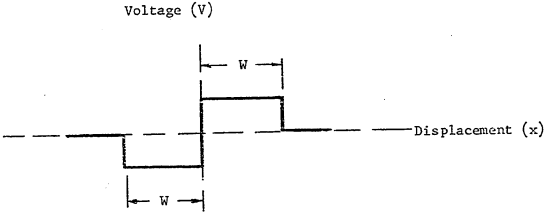


Fig. 2.4

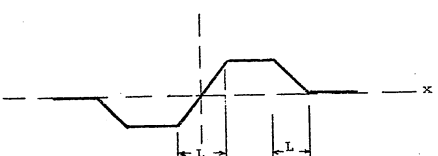


Fig. 2.5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

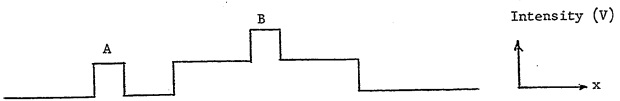
centered at $x = 0$. Looking at the same tracker in a slightly different way, Fig. 2.4 can be considered to be the mask in a correlation function, and Fig. 2.5 would then be the output of the correlation process. The displacement error can be obtained by inverting Fig. 2.5 and determining x directly from the value of V. Unfortunately, the function is only single-valued for $|x| < L$. Further, correct functioning of the tracker requires that the line remain of constant width, that the line never move farther than W away from the center location, that no other objects be closer to the line than $2W$, and that the brightness of the line never change (due either to a change in illumination or a change in the reflectivity of the line). Some of these requirements merely ensure that the gain of the system will remain constant, thus preserving stability. The requirements about spurious objects and about the motion of the line in the field represents a fundamentally damaging tradeoff, however. Basically, one is forced to choose between requiring a large clear area around the line so that the photocells see no other objects, or requiring very accurate small-deviation tracking to get by with small values of "W".

Let us now consider a better mechanism for doing the same job, and then extend the mechanism to draw some conclusions about recognition of simple features.

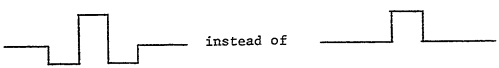
A straightforward way to find a line (which in one-dimensional cross-section appears as a box function) is to convolve it with a box mask function and take the maximum as the location of the line. Although the value of the maximum changes with level shifts and with scale, its location does not, and the location, not the value, is what a controller needs. Also, if the box mask is of width W, objects farther away from

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the line than W will not affect the maximum. Thus the width of the clear space around the line and the total width of the sensor field need no longer be traded off against each other. It should be noted that the maximum referred to must be a local maximum otherwise a 1-D cut such as



would be analyzed to detect B as a cut through a line, but would never find A. Searching for local maxima is equivalent to using a mask function of the form



but the two approaches are otherwise equivalent.

Since both the slope and intercept (equivalent to the displacement rate and displacement) of the line are required to ensure stable control, two 1-D cuts across a 2-D picture are required. This introduces the additional complication of ensuring that the maxima found on the two different cuts are both part of the same line. After all, the picture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

may contain all sorts of additional objects besides the line we are looking for. Fig. 2.6 is an example of such a picture. Here, the blobs marked A, F, and G do not belong to any line, whereas BC and DE are lines, with points B, C, D, and E being the intersections of those lines with the 1-D cuts used for correlation. If the prediction portion of the complete control system were operative, part of the picture would be excluded from correlation because of the impossibility of the guiding feature's occurring there, based on its previous position. If the guiding feature were the line BC, the vertical dotted lines in Fig. 2.6 indicate the area that might be excluded from consideration. Thus the correlation process does not even consider line DE or the points F and G. The point A on the top cut is within the predicted limits, however, and from the 1-D information there is no way of telling whether the guiding feature actually passes through A and C or through B and C. In order to quickly resolve the ambiguity, a number of points between A and C, and between B and C are checked. In order for a line to exist between A and C, the points marked ⊗ would have to be bright, while the points marked "." would have to be dark. Since the "." points are bright, AC can be rejected. When the same test is applied to BC, it passes. This test of a representative sampling of points does not firmly guarantee the connection of B and C, but it is an extremely efficient way of rejecting AC, since only a few points need be considered. If the number of points checked is sufficient, the heuristic is enormously successful, and the computing time saved is well worth the small risk.

But suppose the feature we wish to find is not spatially limited. If we wish to find an edge, we cannot convolve with another edge, because

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

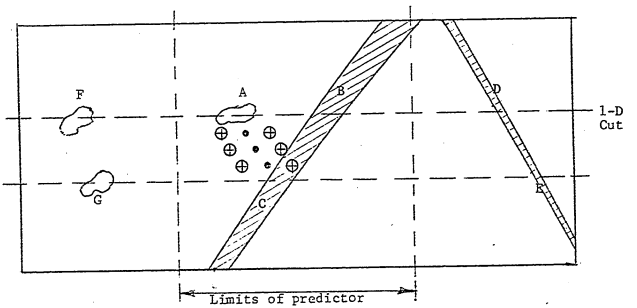
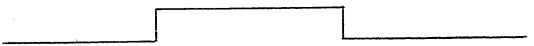


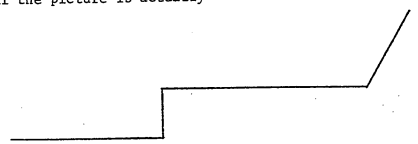
Fig. 2.6

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the integral is unbounded, nor can we convolve with a section of an edge, such as



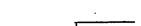
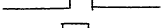
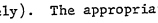
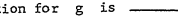
since if the picture is actually



the edge will be improperly identified.

An interesting solution comes from Laplace transform theory. Recalling that multiplication in the s-domain is equivalent to convolution in the t-domain, and that taking a t-derivative is equivalent to multiplying by s, we arrive at the following theorem, which is well known but slightly restated for the present application:

Theorem: If $\int f(x)g(x-t)dx$ is maximized at the desired feature point of $f(x)$ and $F(x) = \int f(x)dx$, then $\int F(x)G(x-t)dt$ will be maximized or minimized at the same point if $G(x) = g'(x)$.

Thus if F is  then f is (approximately). The appropriate function for g is  thus G is  or  to maximize rather than minimize the correlation. Since the integral in the above

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

theorem is over infinite bounds, the general scheme for finding the convolution function to use to isolate any particular guiding feature $F(x)$ is as follows: Compute the n^{-th} derivative of F, such that n is the smallest number resulting in a derivative which is zero except in the region of interest. Convolving this function with itself will obviously result in a maximum at the guiding feature. If this function is differentiated n more times, to produce the function $G(x)$, the integral $\int F(x)G(x-t)dt$ will be maximized at the desired location. Of course, as n becomes large, the effect of noise on F and the effect of approximating G will make the scheme less effective. Even so, it suggests an algorithmic method for computing the convolution mask for an arbitrary guiding feature.

In two dimensions the problem is much the same, except that the mask is two-dimensional and the values may be calculated using gradient techniques rather than derivatives. For computational simplicity, one may as well calculate the mask using the derivatives (differences) taken along the path used by the innermost loop of the 2-D convolution algorithm.

In our actual robot vehicle, the generality discussed here was not implemented. To save time, the convolution masks derived here were hand-coded and the appropriate algorithm was selected by the operator.

C2. The Motion Predictor

The motion predictor is used in conjunction with a wide-field sensor in order to reduce the amount of data which must be analyzed. It uses the equations of motion of the vehicle and the known control inputs to predict the future location of the guiding feature.

The actual equations for the vehicle's motion (only slightly idealized)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

are a system of 12 non-linear equations in 12 unknowns, some of which cannot be controlled or measured by the computer. Various unknown constants also occur, such as the wind velocity, and the road coefficient of friction. In order to simplify this system, the assumption has been made that the vehicle is in effect a slow-speed bicycle. Thus it moves in circular paths such that the front and rear wheels are tangent to the circle. The resulting equations are:

$$V_x = v(\sin \theta)$$

$$V_y = v(\cos \theta)$$

$$\partial \theta / \partial t = v\delta / w$$

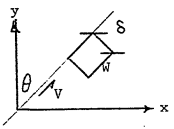


Fig. 2.7

For this particular vehicle it is possible to ignore the effects of centripetal force and linear acceleration, as well as the actuation lags in the controls. For a higher-speed vehicle, the equations would necessarily be more complicated.

To predict the motion in the camera image coordinates, one must also have equations transforming from ground to image coordinates, as well as a function describing the angle δ in terms of the binary output of the computer. The first is straightforward perspective geometry, and the second is a function of the particular hardware control transmission scheme, so neither will be discussed here. See Appendix I for a description of the hardware portion of the control scheme.

The primary errors in the motion predictor, aside from those introduced by the simplification of the equations, are errors in δ, θ and v. Differences between the actual and expected values of δ show up as variations in the rotation rate of the vehicle during turns.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Variations in the camera azimuth produce erroneous measurements of θ, which show up as mistaken predictions of X, Y, and θ. Errors in V show up in both places, but fortunately errors in V are of small percentage value, typically no more than 10 percent, since V is an approximately constant positive number. Both δ and θ can be of either sign, so errors of small magnitude can have quite a large percentage effect.

A record of the past successfulness of prediction is kept in the model corrector, and is used to set the confidence level of the predictions made by the motion predictor. The confidence level is used in the picture analyzer to decide how wide an area around the prediction to analyze. The scheme currently used is to analyze an area as wide as four times the error last time through.

The motion predictor, when used on lines, is able to predict the position of the line to within 10 percent of the full image width, even with large motion of the line in the scene. If used in the same way in two dimensions (circling a point for example), it would thus permit a factor of 100 saving in computation time. Since the entire guiding process takes about 100 milliseconds per iteration, and is repeated once per second, this is the difference between success and failure.

C3. The Control Generator

It should be clear from the motion equations given previously that the vehicle is a $1/s^2$ plant with respect to position control. Thus the control algorithm must include error rate as well as error information. In the photocell tracker described earlier the rate information is obtained by displacing the sensor forward from the center of rotation, thus combining rotation as well as displacement information in the single sensor

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

input. The relative signs and magnitudes of the two terms are fixed by the geometry; in particular, such a vehicle cannot back up using the same sensor, because the rate information will have the wrong sign. In the A.I. Project system, the angle (rate) and the displacement of the vehicle from the desired path are separate outputs of the picture analyzer, and I am therefore able to combine them as I wish.

The actual control used is $\delta = \alpha\theta + \beta(x - x_o)$, where both α and β are functions of the distance moved between picture analyses (x_o is the desired lateral position with respect to the guiding feature). Since $\partial x = \partial s(\sin \theta)$, and $\partial\theta = \partial s\delta/W$, both the error and the error rate increase with both vehicle speed and the time between analyses. Both α and β must be reduced to maintain control. At higher speeds, one must correct position errors with smaller convergence angles to avoid overshoot, implying lower values of β . Due to the discrete nature of the system, it will inevitably oscillate around its path while converging to the desired path, so α must be kept small to avoid large amplitude limit-cycle type oscillations during this process. Further, each term must be bounded in order for convergence to take place with large initial errors. For example, if $\alpha = 1$ degree/degree and $\beta = 5$ degrees/foot, then an initial error of 10 degrees divergence plus 10 feet displacement would produce a 60 degree correction, which would only succeed in driving the vehicle in a small circle. Runs 14 and 15 in Section E of this chapter illustrate both effects in exaggerated form on a simulator.

The other possible benefit of variable parameters is that they allow a tradeoff between steering precision and computing time, lateral

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

acceleration and rapid response.*

04. The Model Corrector

In the current system, the model corrector is the most ad-hoc element. This arises from the previously mentioned excess of errors over error measurements. To illustrate, a steering angle of δ_1 may be expected to produce a rotation rate of $\theta_1 = v_1 * \delta_1 / W$, but instead produces a rotation rate of θ_2. Is this because the actual angle was δ_2, or because the speed was v_2? Similarly, an error in expected position may be due to speed error, initial θ error (due to camera rotation) or an error in either the front or back wheels. Thus one cannot accurately determine the source of error. The original approach was to calculate a new estimate of each error based on the old estimates of all the other errors, and incorporate the estimate into the model with recursive filtering of the form $P = WE + (1-W)P$ with W in the neighborhood of .1. This allows the errors to change slowly, and provided that the initial estimate is not too bad, the system will eventually converge. Unfortunately, there is no way to make a good initial estimate, since the errors are unknown. To improve this situation, the original vehicle was redesigned to have only one pair of steerable wheels, and the wheel steering mechanism was arranged to eliminate drags during turns, which was a prime source of speed variations. This left the camera pointing

*Choosing α and β for sluggish response gives poor tracking performance, but since X and θ change only slowly the control need not be updated as often, resulting in a savings of computer time. On the other hand, very accurate tracking will result in higher lateral accelerations of the vehicle to maintain close displacement tolerances. This might be of importance when changing between highway and urban driving, or when computing time is needed for higher level processes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

mechanism, the steering and the vehicle speed as the three prime sources of errors. The two most significant of these, the camera azimuth and the steering wheel angle were chosen for correction. Speed errors were not corrected, for reasons explained in Section C2. If we assume corrected values for everything but δ and W (camera azimuth), we can calculate δ and W from the measured motion of the vehicle as follows, by inverting the motion formulas.

If $d\theta/dt = V\delta/W$

then $$\theta_f - \theta_i = \int_{t_i}^{t_f} V\delta/W \, dt = (V\delta/W)(t_f - t_i)$$

but if $\delta = \delta + \delta_e$ and $\theta = \theta + \theta_e$ { where δ is the expected value and δ_e is the steering error, and similarly for θ and θ_e

then

$$\theta_f - \theta_i = (V(\delta + \delta_e)/W)(t_f - t_i)$$

or

$$\delta_e = [(W/V)((\theta_f - \theta_i)/(t_f - t_i))] - \delta$$

and

$$\delta = (W/V)((\theta_f - \theta_i)/(t_f - t_i))$$

now using the formula for x coordinate motion given in Fig. 2.7 (assuming that the guiding feature is along the y coordinate, we obtain

$$dX/dt = V \sin \theta(t)$$

so

$$X_f - X_i = V \int_{t_i}^{t_f} \sin[\theta_i + (V\delta t/W)] \, dt$$

$$= (-W/\delta) \cos [\theta_i + (V\delta t/W)] \Big|_{t_i}^{t_f}$$

$$= (-W/\delta) [\cos(\theta_f) - \cos(\theta_i)]$$

now using a trigonometric substitution

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$X_f - X_i = (2W/\delta)\sin((\theta_f + \theta_i)/2)\ \sin((\theta_f - \theta_i)/2)$$

if we recall that δ is already known, and using $\theta = \theta + \theta_e$, we get

$$X_f - X_i = (2W/\delta)\sin((\theta_f + \theta_i + 2\theta_e)/2)\ \sin((\theta_f - \theta_i)/2)$$

For our vehicle $(\theta_f - \theta_i)/2 = V\delta t/(2W)$ has a maximum magnitude of $(0.8)\ (0.5)\ (1.0)/2(3.0) = 0.067$ radian, so the substitution $\sin(x) = x$ can be made without significant error, yielding

$$X_f - X_i = (W/\delta)(\theta_f - \theta_i)\sin((\theta_f + \theta_i + 2\theta_e)/2)$$

Solving for θ_e we get

$$\theta_e = \sin^{-1}[(\delta/W)((X_f - X_i)/(\theta_f - \theta_i))] - ((\theta_f + \theta_i)/2)$$

where all the terms on the right hand side are known.

At this point the two error terms are fed back into the motion predictor by weighting the new error values with the old ones and thus calculating updated values of the errors, as described earlier.

D. Guidance Error Recovery

D1. Sources of recoverable error

We have discussed all of the pseudo-linear aspects of the guidance problem, but there is one eventuality that we have not considered. What happens if the picture analyzer is unable to identify the guiding feature? The absence of the guiding feature will obviously be noticed, but what will the controller do about it? The most obvious response is to stop the vehicle, but it would be desirable to avoid this if possible, and in any event the system must be able to get started up again somehow. There are several possible reasons for the disappearance of the guiding feature from the input picture (besides the obvious one - namely that there isn't any guiding feature anymore). The illumination level of the scene could have changed, so that the guiding feature brightness is no longer within the range accepted

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

by the input hardware. Or perhaps the portion of the guiding feature that the analyzer is considering is obscured. Lastly, the predicted location of the guiding feature might have been in error, a particularly likely situation if the guiding feature has sharp turns in it, which the predictor cannot know about.

Two methods of handling the problem are possible. One is to try all the possible combinations of intensity and screen location in parallel. This way, if the guiding feature is in the field of view at all, one of the combinations will result in a successful analysis, and the controller can use this one for guidance. The trouble is that the computer is a serial device, so we really have to try the combinations one after the other. On the PDP-10, it takes about 10 seconds to do this, which is somewhat too long even for a slow vehicle such as the CART.

The second approach is to try only one combination - (the one that worked last time) and change it only if it fails this time. With this approach, the controller can operate smoothly in areas where the picture intensity does not change and the guiding feature is sharply outlined and smoothly curved. Even in the event of failure, there is time to try a few different combinations of brightness and screen location (about 5 or 6 combinations), before it is time for the next iteration cycle to begin. If the order of trying the other combinations is correct for the particular experimental setup, the right one will often be hit before the time is up, and the vehicle can proceed without interruption. With the CART vehicle, the most common problem is that a shadow has changed the overall intensity of the picture, so the CART control program tries this correction first. The next most common problem is that the shadow is so deep, or the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

light so bright that the TV camera simply cannot resolve that area of the screen at all. In this case, the expected location of the guiding feature is extrapolated up and down the picture, and the picture analyzer looks for a section of the guiding feature above and below the area it had checked first. Only if all this fails does the control program throw out the prediction and analyze the picture as if it had no prior knowledge of the location of the guiding feature.

By now, however, the vehicle will have halted, even if the guiding feature was in exactly the right place. In order to provide fewer interruptions, more time must be provided for analysis before the vehicle halts. This leads us to the last box in Fig. 2.2

D2. Command Enqueuing

Obviously, if one has sufficient confidence in the predictions of the future position of the guiding feature, one need not look to make sure that it is actually there. One can use the predicted location as the basis for control. Even with a somewhat inaccurate prediction the resultant control is better than continuing with the old settings of the actuators. However, there are organizational problems involved in getting part way through the combinational process described in Section D1 and then dropping it and using the prediction instead. For this reason, in the CART system each actual measured value of the guiding feature location is used first to derive control settings, followed by a prediction of the future guiding feature location based on its current location and the new control settings. Then the predicted value is used to compute future control settings, which are in turn used to compute future locations,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and so on. The depth of the process depends on the current accuracy of the predictions. All of the control settings thus obtained are enqueued one after another, and are extracted at regular intervals to actually steer the vehicle. Only when the queue is completely used up does the vehicle halt. In the normal course of events, when analysis is proceeding successfully, only the first element in the queue (the one derived from actual measured guiding feature location) is used. By the time that the second element would be taken from the queue, the entire queue has been replaced by a new queue based on the next measured feature location, so the remaining elements are discarded without being used. Only if the picture analysis takes longer than the iteration interval does the vehicle actually use the predicted values of control settings. These can be good for several seconds of motion, and take very little time to calculate, compared with the time involved in picture analysis. Their effect is to improve system performance dramatically when the guiding feature is fairly straight, but difficult to see. On curves, the predictions are poor, so the queue is short, which is just as well, since the predicted control settings are also poor, being based on a straight line extension of the guiding feature. Section E contains some runs illustrating the effect of the command enqueueing mechanism.

E. Experimental Performance of the CART Guidance System

The guidance system for the CART successfully drives the vehicle at a speed of about 1.2 ft/sec. (0.8 mph) along a path laid out to approximate a white line painted on a road. It also can follow the same path while guided by the parallel edge of the adjacent building. The program acquires the line at the beginning of the run without manual intervention, compensates

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

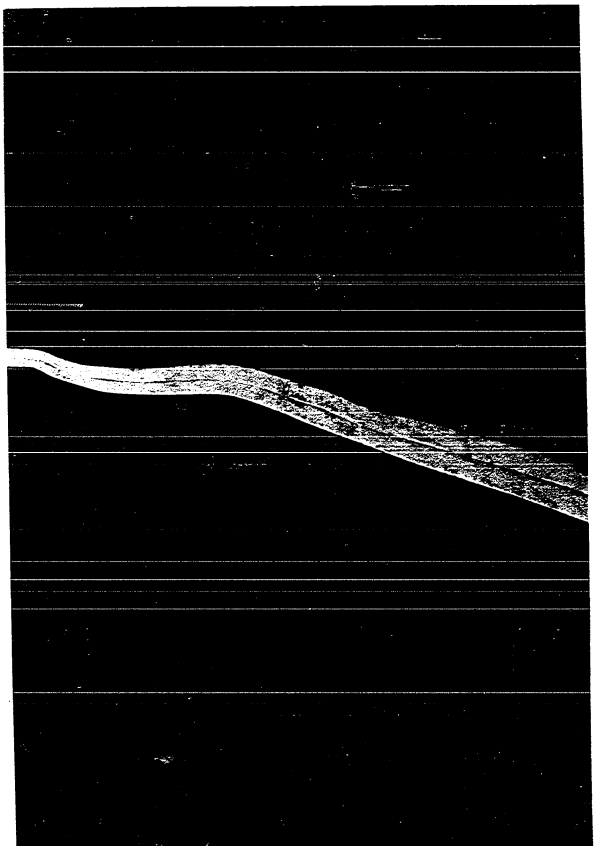


Plate 2 - Taped Line Used for Guidance Experiment

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

for variations in illumination (shading) during the run, and stops when the end of the line is detected. Due to the difficulty of collecting statistics about the actual vehicle, the data given here comes from a simulation, written in ALGOL, which includes most of the control aspects of the actual system. What is omitted is the actual picture analysis, the algorithms for visual accommodation and the failsafe portions of the control, since the picture analysis cannot be conveniently simulated and the fail-safe features do not show up in normal operation. For a discussion of accommodation see Tenenbaum [7].

The experimental conditions simulated are as follows: The vehicle, with a wheelbase of 3 feet and a turning circle of about 12 feet, is moving along a marked path at 1.2 ft/sec. Pictures of the path are taken every second (simulated by a table look-up) - if the vehicle is speeded up, they must be taken more frequently. For simplicity's sake, the simulated runs are drawn with the desired position on top of the marked guiding feature, although the actual program is capable of maintaining any desired offset with respect to the guiding feature. The curve radius and path length are given for each run as an aid to grasping the scale of the presentation - the length of each run is 40 or 50 seconds (if the actual vehicle were to do it - the simulator takes only 1 or 2 seconds).

The picture analyzer returns both the (simulated) perpendicular spacing of the vehicle from the guiding feature and the (simulated) rotation of the vehicle with respect to paralleling the desired course. These two kinds of errors are weighted and combined to determine the angle of the steering wheels during the next iteration interval. The formula is: <steering wheel angle> = <steering displacement sensitivity> *

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

<perpendicular displacement error> + <steering angular sensitivity> * <rotation error>. Obviously this formula cannot be applied literally, since the potential displacement error is unbounded. In the guidance system, an upper bound is placed on the size of the steering angle which may be generated by displacement error. This is the "displacement correction limit" given in the simulated presentations. The hardware places a limit on how sharply the wheels may be cut - this is the "maximum correction". The effect of the maximum correction is only felt on corners sharper than the turning radius of the vehicle, but the displacement correction limit has a much stronger effect. For large displacement errors, the displacement error correction term is held fixed by this limit, and the vehicle rotates toward the guiding feature until the rotation error is large enough to counteract this fixed amount. Thus from large distances, the vehicle approaches the guiding feature from a fixed angle determined by the size of the displacement correction limit and the steering angular sensitivity. This is shown in Run 1.

Steering error and camera error are offsets in the pointing of the wheels and the camera. The controller initially assumes that 0 degrees for the wheels and the camera is straight ahead, but errors in the hardware make this assumption often invalid. Several of the runs show the effect of these errors, and what can be done about them.

"Maximum displacement error" and "Average magnitude of error" are performance statistics gathered for each run on the simulator, and should be self-explanatory. "Average width of field of camera" is somewhat more complex. Basically, if the guiding feature has been located in the camera image field, and a certain course correction is ordered, it is possible

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

to predict where the guiding feature will be next time the program looks, based on an internal model of the vehicle behavior. Since the model is imperfect, and the guiding feature has a shape unknown to the program, the prediction is not perfectly accurate, but it indicates a general area on the image which should be analyzed next time, while ruling out other areas. Since the processing of the visual data is the prime computational task involved in guidance, in terms of elapsed time, even a small reduction here shows up directly in the computational efficiency of the system. The statistic shown in the runs is the average percentage of the visual field which was actually searched during the run - a score of 20 percent indicates a 5 to 1 reduction in computing time for the actual system. In the simulation, the minimum field width that was allowed to be analyzed was 10 percent, in order to avoid narrowing the analyzed field so much that the first curve would put the guiding feature outside the limits of analysis.

In the simulation, on the runs where the internal model was not corrected during the run, predictions were not used either, so they got 100 percent visual field scores. This was unnecessary - predictions from an uncorrected model can be used, and the results are no more than 50 percent worse than for corrected models, depending on the amount of difference between the model and the actual vehicle.

Let me now proceed to a discussion of the actual runs and what they point out about various aspects of the control scheme.

RUN 1 -

This run represents the ideal behavior of the system in correcting a large initial error. The vehicle was started up parallel to the desired track, and 5 feet to the right. The vehicle turned toward the line until

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

a reasonable approach angle was obtained (0.25 radians, to be exact), and then drove toward the desired track in a straight line. When the error became less than 1.67 feet, the vehicle gradually turned to parallel the desired track.

RUN 2 -

With the same initial conditions as RUN 1, but with a 0.1 radian offset to the right in the steering wheels, and a 0.1 radian offset to the left in the camera azimuth, the performance is considerably degraded. These errors, about 5 degrees, are just barely noticeable when looking at the actual vehicle, and maintaining the long term pointing accuracy of the controls to a closer tolerance than this is unusual.

RUN 3 -

Same as RUN 2, but with the corrective and predictive portions of the control system in operation. The error is compensated for the first 10-15 feet of the run, and thereafter convergence occurs normally. The averaging time for error correction has arbitrarily been set at 10 seconds. Less time makes the correction terms too noisy, and more time hurts the ability of the corrector to improve performance on curves (see later runs).

RUN 4 -

Here a curve of constant radius, with zero initial error and perfectly aligned controls. The steady state error is 0.6 feet.

RUN 5 -

The same conditions as RUN 4, except with the corrective and predictive system active. Although there are no actual errors in the controls, the corrective system finds some, based on the expectation that the guiding feature is straight, when in fact it is curved. Thus some corrections are made internally, and the effect is better curve-tracking,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

but for all the wrong reasons. Here the steady state error would go to zero if the curve was long enough.

RUN 6 -

Here is a run with some actual camera and steering errors, and no correction. The performance is quite bad, and for a 6 foot wide car in an 8 foot lane this would result in inability to stay in the correct lane.

RUN 7 -

Same as RUN 6, but with predictive and corrective systems working. Again, steady state error tends to zero.

RUN 8 -

Here we see a complete turn with straight sections at both ends. Performance is not so good, even with perfectly aligned controls.

RUN 9 -

Same as RUN 8, but with predictive and corrective systems running. Here we see the effect of the corrector's misperception of the source of the displacement error. During the turn, the corrector was busily adapting the system to contain a right-turning bias, and when the turn straightened out, the vehicle tracked to the right of the desired path until this bias was adapted back out. Even so, the performance was better than without correction (RUN 8).

RUN 10 -

The same course as RUNS 8 and 9, but with some actual control errors. No correction on this run. The performance is the worst ever, with a maximum error of over 2-1/2 feet.

RUN 11 -

Same conditions as RUN 10, but with prediction and correction. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

maximum error is reduced by over 2:1 and the average error by almost 3:1. It should be noted that the curve in RUNS 8-11 is fairly close to the sharpest that the vehicle can negotiate.

RUN 12 -

This run, taken without correction, shows the effect of control errors uncontaminated by initial errors or curve-following. With the same 5 degree errors as before, the steady-state error is about 1 foot.

RUN 13 -

Here RUN 12 is repeated with the predictive and corrective systems going. The maximum error is about half its uncorrected value, and reasonable convergence is obtained about 20 feet into the run.

RUN 14 -

Here is a run with (effectively) no bound on the displacement correction term. The resulting problem is obvious.

RUN 15 -

This run has a larger than normal angular sensitivity. Although it just barely shows on the displacement plot, the steering is limit-cycling between -0.5 and 0.5 radians every iteration cycle. This creates certain obvious problems.

One of the most interesting items of empirical information which came from the experimental guidance program concerns the magnitude of the computing task. A program of the type used in the CART tests could be implemented on a mini-computer if the application warranted it. The kernel of the control program is only about 4000_8 words long, and might well be implemented on a machine such as the HP2116. Unlike the functions described in later chapters, the amount of TV buffer space required for

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

input data is quite modest - about 30 36-bit words per line of TV data, with only 10 lines of data actually required in core at once. On a 16-bit machine such as the 2116, buffer words would be generated every 600 ns. or so, based on 4-bit samples and a sample rate of 155 ns. Memories of this speed are available, but some sort of shift register arrangement in the input hardware would make it possible to use standard core storage.

On the PDP-10, the basic cycle of steering control takes about 320 ms if no line is present in the input data, and about 180 ms to find one if it is present and prominent. Once the predictor is operating, these figures are reduced by a factor of ten or so. At this point to most significant factor in limiting performance is the delay required to obtain a TV picture from the camera. This delay, which is composed of the hardware delay plus the time-sharing monitor overhead, amounts to between 50 and 250 ms. The actual average delay of the TV is one frame time, or 16.7 ms, so a system operating on a mini-computer would have a significant advantage over the present system, possibly enought to compensate for the lack of such hardware features as floating point and byte manipulation hardware. Byte manipulation is required because of the packed format of the input data from the camera.

Thus this study shows that for some applications it is reasonable to consider implementing the kind of control system discussed in this chapter on a small computer. I am not thinking of automobiles or vehicles exclusively here, but rather more general control problems involving imperfectly known control hardware and moderately complex sensing devices such as TV's. Although this study does not consider the economics of such systems, technically they are perfectly feasible.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This concludes the discussion of guidance. I have presented here the theoretical framework which I believe must be used in pursuing computer vehicle guidance. The hardware and software discussed here are not intended to be finished products, or in any way optimized. They possess sufficient reliability and versatility for our laboratory system to be used to investigate problems in navigation using visual perception. This work, still in its infancy, will be discussed in the next chapter.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 1
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR 5.00 FEET
AVERAGE MAGNITUDE OF ERROR 1.27 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%

RUN 1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 2
SPEED 1.20 FT/SEC. ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .10 RAD, CAMERA ERROR -.10 RAD
MAXIMUM DISPLACEMENT ERROR 5.00 FEET
AVERAGE MAGNITUDE OF ERROR 2.76 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%

RUN 2

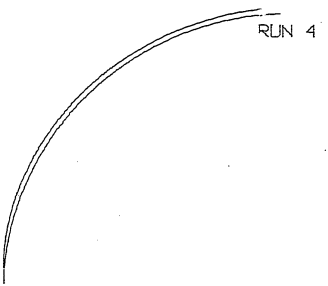
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 3, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .10 RAD, CAMERA ERROR -.10 RAD
MAXIMUM DISPLACEMENT ERROR 5.00 FEET
AVERAGE MAGNITUDE OF ERROR 1.79 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 16.75%

RUN 3

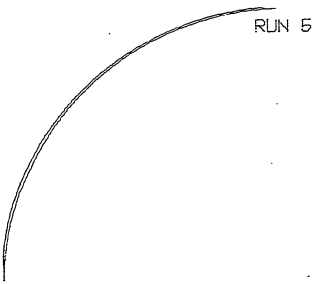
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 4
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 31.34 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR .60 FEET
AVERAGE MAGNITUDE OF ERROR .46 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%



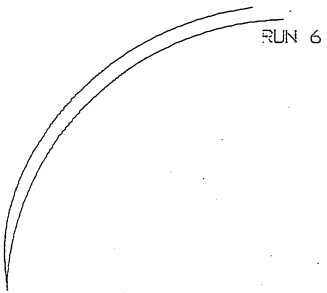
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 5, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 31.34 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR .37 FEET
AVERAGE MAGNITUDE OF ERROR .23 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 14.74



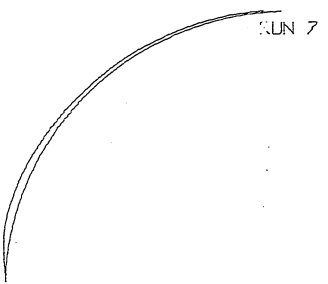
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 6
SPEED 12.0 FT/SEC., ITERATION INVERVAL 1.00 SEC
CURVE RADIUS 31.34 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR -.10 RAD, CAMERA ERROR .10 RAD
MAXIMUM DISPLACEMENT ERROR 1.59 FEET
AVERAGE MAGNITUDE OF ERROR 1.28 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%



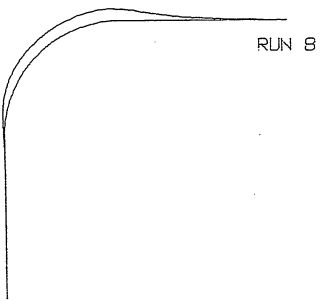
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 7, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 31.34 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR -.10 RAD, CAMERA ERROR .10 RAD
MAXIMUM DISPLACEMENT ERROR .93 FEET
AVERAGE MAGNITUDE OF ERROR .47 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 20.19%



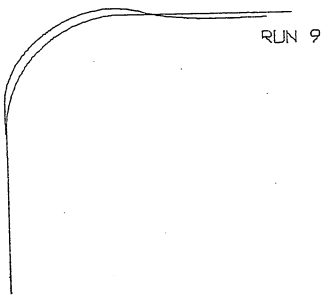
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 8
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 12.99 FEET
PATH LENGTH 58.80 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR 1.36 FEET
AVERAGE MAGNITUDE OF ERROR .50 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%



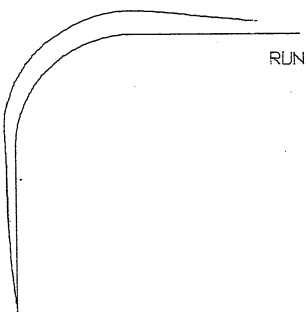
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 9, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 12.99 FEET
PATH LENGTH 58.80 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR .99 FEET
AVERAGE MAGNITUDE OF ERROR .41 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 18.90%



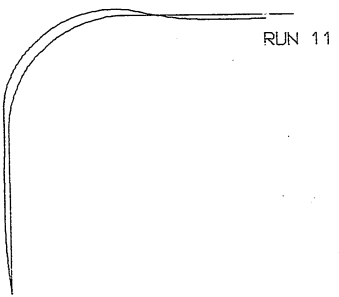
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 10
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 12.99 FEET
PATH LENGTH 58.80 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR -.10 RAD, CAMERA ERROR .10 RAD
MAXIMUM DISPLACEMENT ERROR 2.61 FEET
AVERAGE MAGNITUDE OF ERROR 1.54 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 11, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS 12.99 FEET
PATH LENGTH 58.80 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR -.10 RAD, CAMERA ERROR .10 RAD
MAXIMUM DISPLACEMENT ERROR 1.15 FEET
AVERAGE MAGNITUDE OF ERROR .59 FEET
AVERAGE WIDTH OF FIELD OF ERROR 21.59%



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 12
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR -.10 RAD, CAMERA ERROR .10 RAD
MAXIMUM DISPLACEMENT ERROR 1.00 FEET
AVERAGE MAGNITUDE OF ERROR .81 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 100.00%

RUN 12

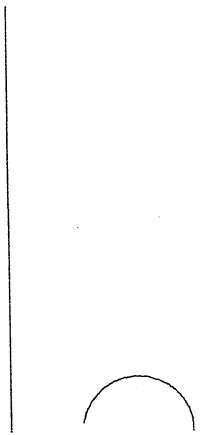
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 13, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT .50 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR -.10 RAD, CAMERA ERROR .10 RAD
MAXIMUM DISPLACEMENT ERROR .57 FEET
AVERAGE MAGNITUDE OF ERROR .23 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 16.77%

RUN 13

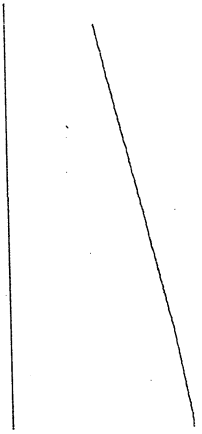
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 14, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD/FT
STEERING ANGULAR SENSITIVITY -2.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT 100.00 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR 20.00 FEET
AVERAGE MAGNITUDE OF ERROR 5.69 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 12.84%



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

RUN 15, WITH CORRECTION
SPEED 1.20 FT/SEC., ITERATION INTERVAL 1.00 SEC
CURVE RADIUS .00 FEET
PATH LENGTH 49.20 FEET
STEERING DISPLACEMENT SENSITIVITY -.30 RAD.
STEERING ANGULAR SENSITIVITY -10.00 RAD/RAD
DISPLACEMENT CORRECTION LIMIT 2.00 RAD
MAXIMUM CORRECTION .50 RAD
STEERING ERROR .00 RAD, CAMERA ERROR .00 RAD
MAXIMUM DISPLACEMENT ERROR 20.00 FEET
AVERAGE MAGNITUDE OF ERROR 14.42 FEET
AVERAGE WIDTH OF FIELD OF CAMERA 14.74



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER III

NAVIGATION

A. Introduction

In the previous chapter, I discussed the design of the CART guidance system. A guidance capability by itself is not sufficient to constitute an automization of the driving process. In order to complete a journey, a navigation capability must be provided for making decisions as to the sequence of paths which must be followed to reach the destination, determining the points (intersections) at which these paths join, and making the appropriate decisions and maneuvers to change paths. The process of choosing the sequence of paths (the "route") is not considered here, since it can be done in advance of a journey.

If one wished to treat the navigation problem as a question-and-answer game, the sequence would go something like this:

Q1: What should I do?

Q2: Well, where are you?

A2: I am at location "X".

A1: Then you should do "Y".

This chapter concentrates on answering question 2 by the recognition and processing of visual images. The capability of using visual images to provide information as to location is critical to the type of vehicle guidance of interest here. The navigational problem is computationally different from the guidance problem, in that it involves discrete decisions (guidance is fundamentally a continuous process) and requires the manipulation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of larger quantities of information. A method of recognizing and processing scenes is developed here which takes cognizance of the fact that two images of the same physical scene may "look" quite different. The problem is dealt with so that the equivalence of scenes can be recognized, and not merely the identicality of images.

Let me begin the discussion with an overview of navigation techniques and of visual image description. Section B of this chapter will then discuss the problems of analyzing and describing scenes, with emphasis upon the problems introduced by allowing the possibility of different images of the same scene. Section C deals with a mechanism for testing the equivalency of structurally and parametrically different descriptions which potentially refer to the same scene. Section D shows some experimental results from a program which actually carries out these functions, and Section E outlines a scheme whereby the results of the scene identification process might be used to make navigational decisions.

A1. Navigation Techniques

Navigation, in some form, is an inescapable part of any journey, if for no other purpose than determining when the destination has been reached. The actual implementation of navigation can be done in any of three ways (or combinations of them):

1) To have surveyed the journey space very carefully, thus obtaining the relationship of the intersections to each other, and to then calculate one's motion (and hence position) from the control commands issued or from measurement of vehicle acceleration (otherwise known as dead reckoning).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

2) To establish an artificial reference frame, perceivable by special instruments, and then note the position of the intersections with respect to the reference frame. This is the method used by such systems as omnirange, DECCA, and LORAN.

3) To determine the intersection by perceiving directly the meeting of paths at that point, and/or whatever else naturally occurs there. The major advantage of this method, called pilotage, is that it requires neither prior measurement nor external equipment. The corresponding drawback is that the recognition of naturally occurring intersection characteristics is usually much more difficult than detection of specially installed devices.

Of these three methods, pilotage is the most desirable for surface vehicle navigation. Accuracy is a problem with dead reckoning, as well as a reliance upon unchanging positions. External systems are necessarily large in scale, and therefore expensive. Hardware used in such systems is usually specialized, and contributes little to the solution of problems of incident avoidance. In addition, the man-machine interface becomes additionally complex in such systems, since the machine recognizes landmarks from cues neither apparent nor epistemologically significant to a human operator.

A2. Navigation by Pilotage

If one is going to navigate by pilotage (the recognition of naturally occurring characteristics), one must "remember" information sufficient to identify these characteristics. Since the characteristics are generally complex, the information will be non-trivial. The "remembered" information

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is called a "description" of the characteristics, and may be stored or processed in several different ways. I concentrate on the description of visual images of things because they seem the most valuable, but the epistemological considerations are similar for other sensory images.

Basically, one can choose to describe reality on any of three levels, corresponding to the three levels of human consciousness. On the sensory level, a human's view of reality is a matrix of color and intensity information derived from the cells in his retina. The digitized image from a TV camera is a similar image, although of lower resolution, dynamic range, and deficient in color information. A characteristic of this image is a high ratio of data to information. In addition, the information is unorganized, and is so loosely distributed through the data that extracting any particular item from it is very time-consuming. For this reason, computer programs functioning at this sensory level almost never store the sense data, and work well only in cases when the data is very predictable. The line follower of the last chapter was such a program.

The human brain almost never functions on this level. Instead, man's perceptual facility automatically combines the data and presents it to one's conscious awareness as shapes possessing homogeneous properties. The exact form of the presentation is influenced by the purpose of the viewer (which accounts for many "optical illusions"). The same process is carried on by computer programs which attempt to extract information from the sense data by locating either regions of constant parameters or areas of rapid parameter change ("edges"), and then preserving only the outline of these features. (If you think this process is easy, since you do it all

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the time, try looking at a room through a pinhole, scanning the pinhole over the scene. This forces you to perform consciously the perceptual integration that your brain normally does automatically, and you will find it almost impossible to figure out what you are looking at.)

The third level at which description is possible is the conceptual level, where items are characterized by what they are and what properties they possess. At this level, the image of the object is no longer the only information used, and the description may no longer have a unique image associated with it, i.e., knowing that an object is a "house" does not enable one to draw a picture of it. In a human, this level of description is reached by combining the perceptual image with prior perceptual and conceptual knowledge to make a correct identification. The prior knowledge may well come from some other source than visual images. Computer programs which attempt to duplicate this level of description have not fared very well due to the lack of multi-sensory data and the tremendous amount of processing involved in even the simplest sensory integration.

The work described here is an attempt to combine some of the best features of the perceptual and conceptual descriptions. The necessity for this compromise approach arises from the horns of the following dilemma:

1) In a perceptual description, small changes in the visual properties of the viewed object (call it the "target") or small changes in the view angle or illumination, can cause large changes in the description. Especially annoying is the case where the target is a complex scene, and the boundaries are shifted enough to move objects or parts of them out of view.

2) A conceptual description requires information not present in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the image currently viewed, and a mechanism for integrating this information in a fairly general way with the perceived data. Doing this kind of processing in real time, although necessary for incident avoidance, is completely beyond the range of current processing techniques.

The compromise I adopted was to describe images in terms of semi-invariant properties of the perceived shapes in the image. This information is stored in such a way that the various subelements are independent, thus curing the boundary-shift problem. In addition, the properties stored are simple linear geometric functions of the shapes, so that small shape changes produce small changes in the properties; and the properties are simply computable, so that the processing time is much smaller than for a direct comparison of outlines.

I will proceed now to the detailed description of the method I developed for the experimental vehicle, and a discussion of picture properties that make this method a computationally efficient, but imprecise, way of perceptually describing scenes.

B. Picture Description Techniques and Algorithms

A "picture" is a two-dimensional array of points, potentially separated by discontinuities in those characteristics. The characteristics themselves may be any measureable property of the scene, such as intensity, color, or texture. In the work described here, the intensity of a local region about each point was used as the characteristic. This choice was made primarily because of the high computational speed required. To sense color, three pictures must be taken through appropriate colored filters, and the color information extracted. This is a relatively slow process since it must process three times the data as a B and W picture, and must

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

normalize out intensity variations. Further, the vehicle is in motion during the sequence of pictures, and the compensation for this is non-trivial. Texture information was not used because algorithms for texture extraction are just now being developed and no information about the general usefulness of this technique is yet available. Intensity information is available at low computational cost, and appears adequate for the time being. There is no doubt, however, that if the processing difficulties could be resolved, color information would make an extremely significant improvement in perceptual ability.

The structure of the recognition program is considerably different from the structure of the guidance program, so the format of this chapter is different from that of Chapter I. Most of the detail of the recognition system is internal bookkeeping, so the discussion in this chapter will focus on design concepts and tradeoffs. A block diagram of the recognition system is given in Fig. 3.1. In this figure, processes are in rectangular boxes, while data descriptions are in circles. Although the various steps are presented as if they occurred sequentially, in the actual program they are interleaved so that if the entire picture does not have to go through the parsing step before a decision is reached, the remainder of the picture does not undergo any analysis. In some cases this can save a large amount of processing time.

B1. Region and Edge Operators

Once we have a field of characteristics, however chosen, there are two fundamental ways to go about describing it. We can either note those areas in which the characteristics change dramatically and record a description of those areas, or we can note the areas in which the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

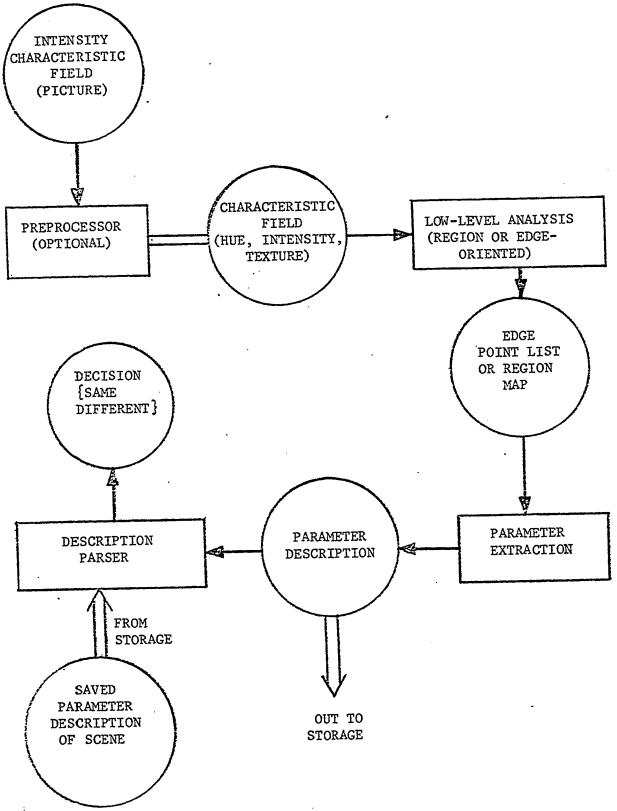


Fig. 3.1 - Scene Analysis Flow Diagram

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

characteristics remain constant, and record their description. This is the choice between "edge operators" and "homogeneity operators" * The CART system uses a homogeneity operator, and in the next few pages I will present several examples which show the performance differences of the two operators which led me to select this approach. Bear in mind the two goals of the description process: 1) Small changes in picture result in small changes in description - changes of coefficients rather than changes in the description structure. 2) The description process must not introduce any artifacts into the picture, such as corners that aren't really there, or parameters dependent upon particular idealizations of objects, or anything else that might not be the same in a slightly different version of the same scene.

Consider the following scene:

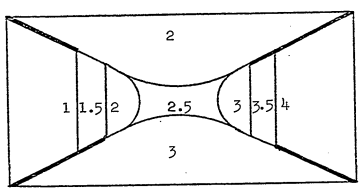


Fig. 3.2

The numbers represent the intensity of the various areas of the picture. This picture is composed of essentially four regions, with the intersection

*In this context, an "operator" is an algorithm which analyzes small local areas in a picture. The operator is scanned or "flown" over the picture to determine the location of whatever local property it is designed to detect.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

blurred for some reason. An edge operator with unit threshold (i.e., one which locates discontinuities of 1 or larger) would find edge points as indicated by the heavy lines. A different picture, such as:

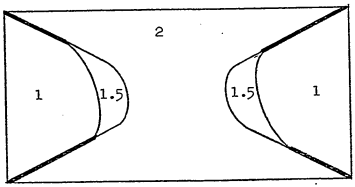


Fig. 3.3

would result in the same edges. Obviously the two pictures are structurally quite different, but the decision procedure to extend the lines in the first picture, but not in the second, would be quite involved. On the other hand a homogeneity operator with unit threshold (i.e. one which accepts points within 1 unit of the current average value) analyses the first picture as:

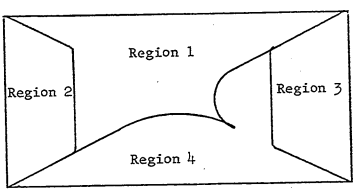


Fig. 3.4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

or

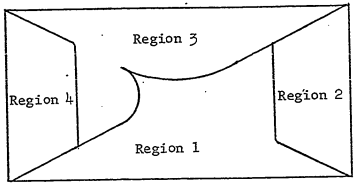


Fig. 3.5

depending on exactly how it got started.* The second picture would be analyzed as:

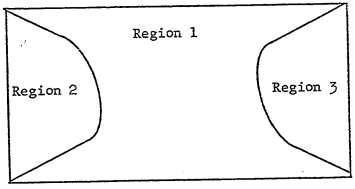


Fig. 3.6

In this case, the boundaries are slightly distorted, but the essential structure of the picture is preserved.

Another nice feature about homogeneity operators is that in the scene, regions of like characteristics belong to the same object. In an edge description, objects which overlap or occlude others have their edges described inseparably. In particular, objects next to visually

*There are other possible outcomes but all are basically similar to either Fig. 3.4 or Fig. 3.5.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

complex objects become complex themselves. In a homogeneous region description, boundary redundancy is provided, and only the complex region is lost. Consider the ordinary visual acuity chart shown in Fig. 3.7:

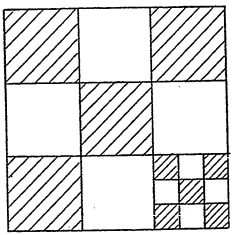


Fig. 3.7

An edge operator would find the lower right corner to contain a large number of disjoint, random edge points. Likewise a region operator would find a large number of region nuceil, too small to describe. The difference is that the adjoining white squares would be well defined although with somewhat ragged edges. The edge operator would have difficulty extending lines along those boundaries. In addition, the outer structure of the complex small checkerboard is potentially still recoverable from the well-defined edges of the white squares, even if the internal structure is too complex to handle.

The one significant advantage that the edge operator has over the homogeneity operator is in its treatment of lines. A "line" is a region which extends in only one direction. In the perpendicular direction, its width is only a few picture elements. Under these circumstances, an edge operator will find a large number of edge points all lined

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

up, and will have no difficulty defining the line. A homogeneity operator may have difficulty outlining the region, since it is not wide enough to be encircled. Consider the following thin annulus:

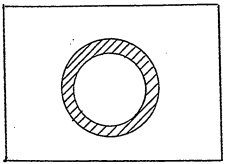
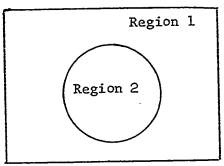
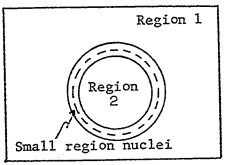


Fig. 3.8

and the results of processing by the two operators:



Edge operator



Homogeneity operator

Fig. 3.9

Notice that though both clearly separate the inside from the outside, the homogeneity operator does not preserve the annulus as a distinct object. Of course, the edge operator does not preserve the annulus as an annulus, that is, possessing an inside edge and an outside edge, but at least it retains the outline. An even worse situation occurs when the annulus has a weak spot. In such a case, the edge operator merely misses a point or

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

two which are easily filled in. The homogeneity operator loses the separation between the inside and outside of the annulus, a profound structural change, shown below:

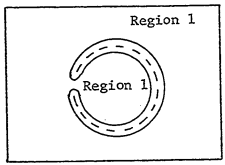


Fig. 3.10

The prime reason for my choice of homogeneity over edge operators is that outdoor scenes contain few fine lines, and do contain many areas of irregular intensity, such as tree shadows and the like.

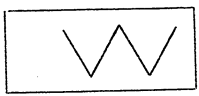
B2. Structure and Parameterization of Pictures

Let me make clear the function which the picture classifier must serve, and the conditions under which it must operate. In order to navigate by pilotage, it must save enough information about intersections that it can "recognize" a previously seen scene and thus determine the vehicle's location. It must do this in spite of translation, rotation, shifting of objects on or off of the scene edges, and moderate change in the objects themselves due to illumination or view angle shifts. The whole process of recognition must take only a few seconds, if it is to control a vehicle in real-time.

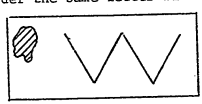
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

At this point, the significance of the dilemma mentioned earlier becomes clear: If the description is a structured description whose layout depends on the nature of the scene, there will be unavoidable singularities in the scene which cause the description to "snap" from one format to another. It then becomes difficult to compare scenes because of the necessity to make comparisons between differently structured data sets. On the other hand, if no structure is used in the description, the various parts of the scene will not maintain a separate identity in the description, and allowance for objects shifted off of the picture boundaries cannot be made. Let me give a simple example from the class of character recognition problems:

Consider the letter "W", as shown below:



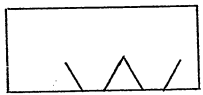
One conventional way of describing letters is by a list of their moments - an unstructured description, since the same moments are calculated for all letters. Now consider the same letter with a smudge in the frame:



The letter is unaffected, yet the description of this scene is quite different than for the first "W", because the mass of the smudge changes all the moments. A structured description would have the "W" and the smudge described separately, so the letter would still be the same. However, this

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is not entirely without pitfalls, as may be seen from the next version of the same letter:



Here, improper positioning of the letter in the scene has chopped the bottom of the letter. The moment description is not much changed, since only a few points were lost, but the structured description is completely different. The letter has been broken into three pieces, a backslash, an "and" sign, and a regular slash. No descriptive parameters of these are the same as those for the "W", and identifying the letter would be a very difficult undertaking.

Let me recast the problem in a somewhat more elegant way. Suppose pictures are regarded as points in an n-dimensional "picture property space". Some areas of this space will be vacant, because it would be physically impossible for a picture to have certain combinations of properties. If we focus our attention upon a particular class of scenes, for example road scenes, other areas of the space will be sparsely populated, because road scenes do not typically contain certain combinations of features. If we introduce a structured description format, the space will be partioned into regions corresponding to the various potential structures (not necessarily a 1-1 correspondence).

If the structure scheme we pick results in the partition boundaries running through areas which contain many pictures, we are going to be in trouble. If small changes in the pictures produce small motions in the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

picture property space, we are going to have a lot of closely related, virtually identical pictures which fall into different partitions of space. Since the descriptions associated with different partitions are different, it may be quite difficult to recognize later, while looking at the descriptions, that the pictures were almost the same. It therefore behooves us to pick our structures such that the partition boundaries run through sparsely populated regions of the space.

On the other hand, if we reduce the variety of permissible structures, and so reduce the number of boundaries in the space, we increase the number of pictures in each partition. It then becomes difficult to derive a sufficient number of parameters to catagorize the members of each partition so that we can tell them apart.

Speaking loosely, the most practical approach is to partition the space as finely as one can without running too many boundaries through densely populated areas of the property space. The comparison problems across these boundaries can then be alleviated by appropriate modifications of the structures associated with the two neighboring partitions. In particular, each structure can contain parameters associated with the neighboring structure, so that limited comparison with descriptions using the neighboring structure is still possible.

An example will help to illustrate the problem. Suppose the input picture was known to contain only n-sided convex polygons. Suppose further that these polygons were approximately regular, so that a circumscribed circle touched all their vertices. Then for large n, the polygons would approximate a circular outline. For small n, an adequate description of these figures could be obtained by listing the edge lengths

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and vertex angles in order around the polygon. For large n, the angles would all be close to 180° and the legs would be short. If telling the difference between a case where $n = 10$ and $n = 11$ were not important, one might choose to approximate all polygons with $n \geq 10$ by circles, and only save the radius.

This would partition the space of polygons as shown in Fig. 3.11. Note that the partition boundary runs through a populated area (if all polygons are equally likely). Now if a polygon with $n = 9$ occurs in our hypothetical picture matching problem, we may have a problem. It is possible that this really is a polygon with $n = 9$, but it is also possible that only a slight change in its appearance will cause us to classify it as a polygon with $n = 10$, and describe it differently. If this happened, it would be very difficult later to determine that the circle of radius r (corresponding to $n = 10$) and the set of edges and angles $(e_1,\ldots,e_9,a_1,\ldots,a_9)$ in fact described almost the same physical structure.

There are two ways around the problem. If it is known that polygons with $n = 8, 9, 10, 11, 12$ do not occur, then the partition boundary used above causes no trouble. This case is shown in Fig. 3.12. Here a very large change in the polygon structure is required to cross the partition boundary. The object of this approach is to pick partition boundaries such that they run through regions describing pictures not expected in the input set. This approach is not always possible however, and for these cases, alternate methods can be used.

For the example of Fig. 3.11, this alternate is to describe all polygons by the radius of the circumscribed circle, and for $n < 10$,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

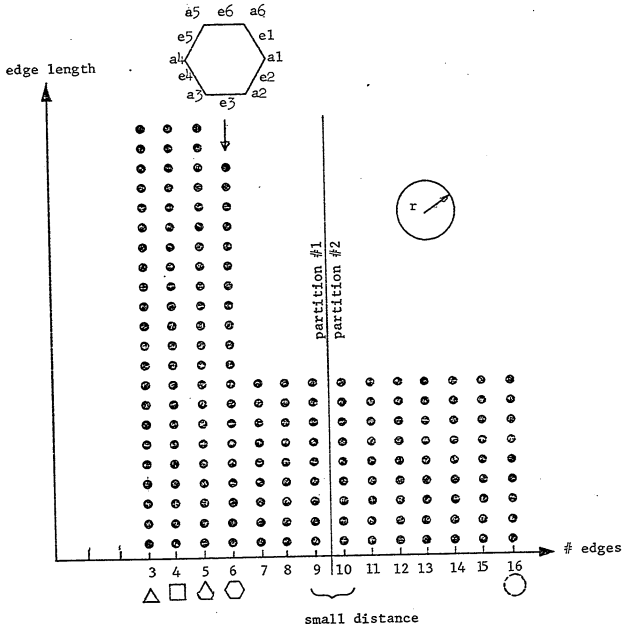


Fig. 3.11

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

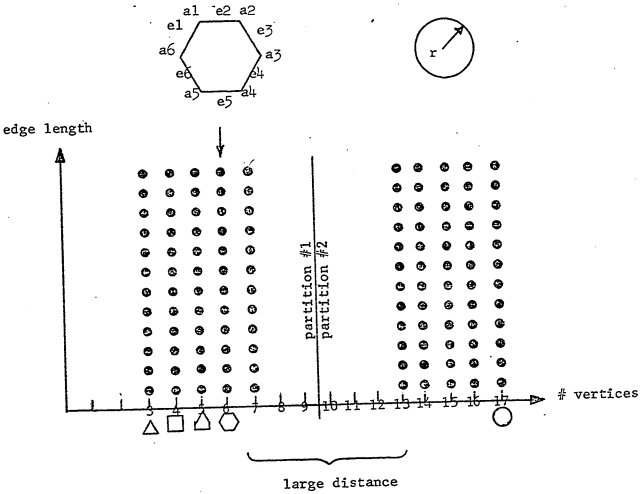


Fig. 3.12

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

also include the set of edges and angles (ej,aj). Thus for polygons with n≥10, the comparison of circle radius can be made independent of the existence of the edge-and-angle structure. Both of these are incorporated in the work described in this section.

Section B2.1 is a discussion of the structural partition which I found appropriate for the analysis of complex scenes. Section B2.2 deals with the various sorts of parameters which I found useful to characterize pictures, and Section B2.3 discusses the redundancy of parameters and structures which I introduced to facilitate cross-structural comparisons.

B2.1 Structure

The scene description I chose was a semi-structured format, in which the various separate objects were described separately, but in a redundant fashion, so that a limited amount of comparison across structures could be done. Hopefully, there will be enough objects in a picture that loss of a few at the picture boundaries will not hurt too much. On the other hand, it is certain that all views of a scene will not always contain all the same objects, so some structure is essential. I make the following structural distinctions between objects:

small versus large -

This somewhat arbitrary distinction is used to decide whether to include a straight-line approximation of the outline of a region (I use "object" and "region" interchangeably here.) in the description. The approximation of the outline of a small region would change too much from view to view to be useful because the region size is insufficiently larger then the picture resolution.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

simple versus complex -

No decision is involved here, merely a distinction between regions bounded by a single exterior curve and regions with interior excluded regions (containing other regions). The picture background is an example of a complex region, since the various objects in the foreground are regions excluded from the background. If the background can be detected, it can simplify analysis considerably, since the background contains more boundary points than any other region, and takes processing time accordingly.

total versus partial -

The exterior boundary of a total region does not touch the picture boundary. Thus its shape is invariant with coordinate shifts, and global measures of its shape can be used to describe it. A partial region touches the picture boundary, so its true shape is unknown. It can only be described in terms of local features of its shape, such as corners, since the global properties change with the picture boundary. Of course, a picture consists of a number of objects, but the object, or region is the fundamental unit of comparison. Thus the variation of picture description structure caused by changes in the number of objects creates no partitioning problem. Only changes of the description of a region itself create major comparison problems. The structural properties of regions will be abbreviated in the discussion which follows. Thus an SSTR is a Small Simple Total Region, an LCPR is a Large Complex Partial Region, and an SCTR is a Small Complex Total Region.

B2.2 Parameterization

Once the structure of the picture description is determined by the topology and size of the picture, one must decide what kinds of information

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

about the picture to hang on the various nodes of the description. The following general criteria for picture parameters serve to guide this process. First, the parameters should be as orthogonal as possible, for there is no point in saving many numbers all describing the same feature. Second, the parameters must be relatively invariant with respect to the non-reproducible (noisy) aspects of the picture, and vary strongly with significant changes in the picture. Third (and last), they must be computationally (and not merely conceptually) easy to calculate, and must be expressible in a form that does not heavily depend upon a particular coordination of the picture.

After considering these issues, I decided upon the following set of parameters. They are not intended to be exhaustive, but they seem sufficient for the purpose at hand.

For small regions, the parameters are the region intensity, the area, the center of gravity, the radius of the largest inscribed circle, the radius of the smallest circumscribed circle (corresponding to the major and minor axis radii), and the locations of the points defining those radii (which gives a measure of the orientation of the major and minor axes).

For large regions, all of the information associated with small regions is saved, as well as a list of points roughly outlining the region (primarily for the human operator's benefit) and a list of prominent angularities (vertices) on the region boundary, including their location and the value of the angle. The details of this are in the section on "redundancy".

Obviously, not all this information is useful for all kinds of regions. The c.g. of a partial region will change as the picture boundary

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

includes more or less of the region. If it is possible to tell that a particular item is useless at the time it is being calculated, the calculation is omitted to save time, although storage space for the item is always provided (except for the point and vertex lists, which are dynamically allocated).

B2.3 Redundancy Mechanisms in the Description

When pictures are described in this way, the comparison mechanism must allow for changes in description structure produced by small changes in the scene or its boundaries. In the case of regions, it must allow for changes of region type, and for vertices, it must cope with variations about the threshold of "prominence" as used above. The following is a description of the various transformations that may occur, and an explanation of the way they are treated by the CART program.

B2.3a Size

Size variations can cause a region to be ignored, to be described by global parameters, or to be described by globals plus an outline and a vertex list. Call the lower boundary N1; regions with fewer than N1 points get ignored. Call the upper boundary N2; regions with more than N2 points get an outline and a vertex list. N1 is fairly small, about 10. N2 is larger, about 100. The disappearance of regions with fewer than 2*N1 points is tolerated, thus allowing variation at the low end of the size range. At the high end, the absence of vertices on a description with fewer than N2 points is also allowed. Thus if the current image is a square, and its corners are detected ($N \geq N2$), the absence of corners in the stored image will be ignored if the number of points (called M) in the corresponding region of the stored description

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

is less than N2. If $M \geq N2$, the corresponding region will not be considered a match. Similarly, if $N < N2$ but $M \geq N2$, the presence of vertices in the description will be ignored.

R2.3b Vertices

The description and matching of vertices is handled by a fairly complex algorithm whose purpose is to ensure that only corners really present in the input data are described. Since the vertices are found in a line-segment approximation to the input region, small vertices may have been introduced by the approximation process, and it is essential that such "dirty fingermarks" not make their way into the final description. Accordingly, vertices are tested for location, and vertices too close to the picture edge are discarded, since they might have been produced by the cutoff of a region by the boundary. Angles which are too oblique are excluded, as well as angles defined by very short line segments. An intermediate group of angles are included if the segments describing them are fairly long, but are otherwise excluded. To facilitate comparison; the length of the shortest segment defining a given vertex is included in the vertex description, along with the vertex location. Upon comparison, the disappearance of vertices close to the obliqueness or segment length thresholds is forgiven. The graph of description thresholds and comparison thresholds appears below.

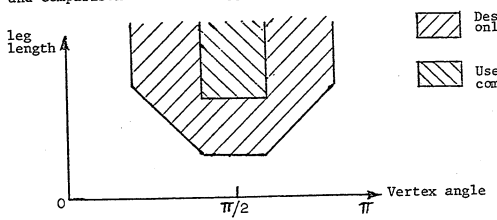


Fig. 3.13

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

B2.3c Picture Boundary

After two objects have been matched in two pictures, the relative orientation of the two pictures can be determined. If the orientation is such that the corresponding region to some object is outside the picture boundary, its absence from the description is (charitably) forgiven. Similarly, if the corresponding region is partial (i.e., hits the boundary), disparity in the global parameters is forgiven. Before the orientation is determined, absence of corresponding regions is forgiven, since there is no way to tell if the reason is picture mismatch or merely a boundary shift that excluded the particular object or region in question. This has a somewhat embarrassing result when two totally dissimilar pictures are compared, in that the program's conclusion is that the pictures match, but have nothing in common! In this case, the relative orientation of the two (unrelated) pictures is never determined, so it is never possible to insist that any particular region be matched by something in the description. Of course, this is readily detectable as a special case, so no operational problems result. Not so trivial is the case when one common object is found in the two pictures, but the remaining objects are dissimilar. In such a case, the program reports "success", even when it would be geometrically impossible for all of the objects to be missing from the description. The examples of tree matching given later in this chapter illustrate such a case.

C. Picture Parsing

Having covered the description process, and the low-level redundancy mechanisms, let me proceed now to a discussion of the high-level strategy

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of the picture-matching process. Fundamental to this process is the idea that the measures of region parameters are relatively crude, both to gain speed, and because precise information is not in the raw data. Confidence is gained in the equivalence of a picture and a description from the relation of a large number of items to each other, and in the rough equivalence of many measures, rather than exact equivalence of a few.

Accordingly, the equivalence set of a region is likely to be greater than one (in the worst case, such as the visual acuity chart, several regions may be identical) and at the beginning of the matching process the number of possible ways that two pictures may correspond is quite large. As the number of regions analyzed grows, the uncertainty should decrease, but the matching mechanism must cope with it somehow, preferably not by pursuing each permutation separately from the beginning.

In the CART program, the set of matches between regions is organized as a tree, whose roots are extended by the analysis of yet another region. The regions in the input picture are matched as they are outlined, with picture reduction, description and comparison all being interleaved. The m nodes of the tree at the nth level are the m possible regions in the description which correspond to the nth region in the new picture. If m = 0, the nth region is discarded, and the level is used for the n + 1th region. The tree is grown recursively, and when prior analysis demands that a match must exist to extend some node but no match is found, that node is pruned, along with any of its predecessors which are no longer viable. Several hypothetical examples follow, which should illustrate the method.

Let me begin with an example of the "proper" functioning of the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

program. (The pictures shown are illustrative only, and may be processed slightly differently by the actual program. In particular, I assume an order in the discovery of regions which will vary in the actual program depending on scene orientation).

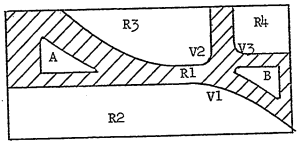


Fig. 3.14 - SCENE 1

This picture consists of 6 regions (3 LSPR, 1 LCPR, 2 SSTR). Fig. 3.15 shows the format used to describe these regions along with pertinent data. The first two cells in the region block are internal bookkeeping, the "region intensity" is the average parameter value found by the subroutine which found this homogeneous region.

The first two cells in the curve block are also bookkeeping, as is the last cell. The "total number of points" is the number of edge points in the region. The "sense" in this same cell tells whether this curve includes or excludes area. Since every edge is described twice, once for its interior boundary and once for the exterior, this is handy to distinguish between the two. The "c.g." in the 4th cell is the average x and y values of all the region points. The radii in cells 5 and 6 are a measure of the shape of the object (for simple regions at least). If the ratio of these is large, the object is long and skinny. In this event, the locations of the tangency points given in cells 7 and 8 will provide information about

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

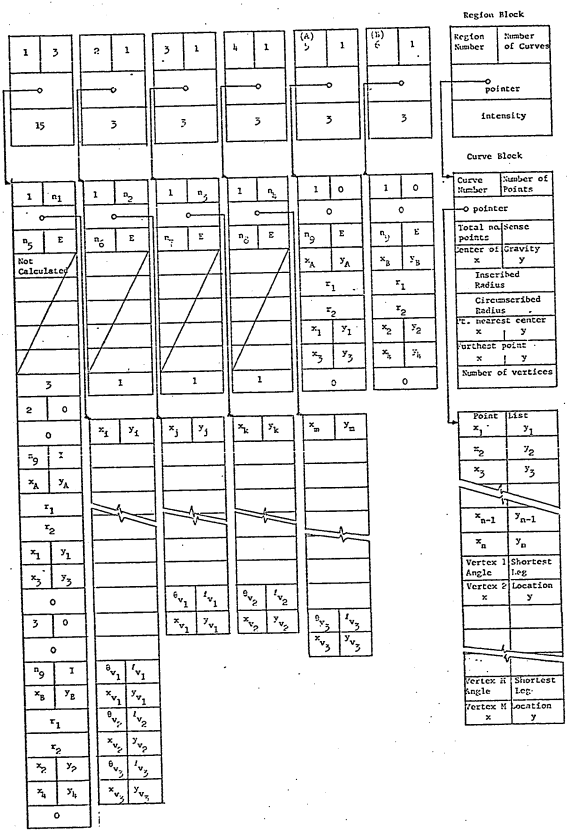


Fig. 3.15 - SCENE 1 Description

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the orientation of the object's major and minor axes, although these may change with only small changes in the object.

Suppose we are confronted with a different view of the same scene, shown in Fig. 3.16, where region 5 corresponds to identical regions A and B in SCENE 1. The rest of the regions are labelled equivalently in the two scenes. Let us follow this picture through the description and matching process and see how the correct identification is made.

Region 1 is parsed first. Its outside boundary is a PR, so no comparison is made. Region 5 is detected as being interior to Region 1, and is a TR, so a search for matching regions in SCENE 1 is conducted. Regions A and B in SCENE 1 are found to match it. A comparison tree denoting this fact is begun at this point. At the same time, the description of SCENE 2 is begun. The two data structures at this point are as shown in Fig. 3.17.

Note that the descriptions of A and B as simple total regions do not appear, since they have the wrong sense. The comparison tree contains the relative translation of SCENE 1 and SCENE 2 appropriate to the correspondence of 1 and A and 1 and B. Although the description contains all the information about vertices V1, V2 and V3, this information is not used for comparison until the end of the comparison process. This is because less information is available about vertices, resulting in a higher tree branching factor as well as a greater chance of error. Proceeding, Regions 2, 3, and 4 are parsed, and the vertex information stored, but no comparisons occur, since 2, 3, and 4 are partial regions. Finally Region 5 is parsed, and is matched with the total region descriptions of Regions A and B. The rotation cannot be determined yet, since

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

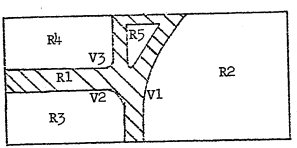


Fig. 3.16 - SCENE 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

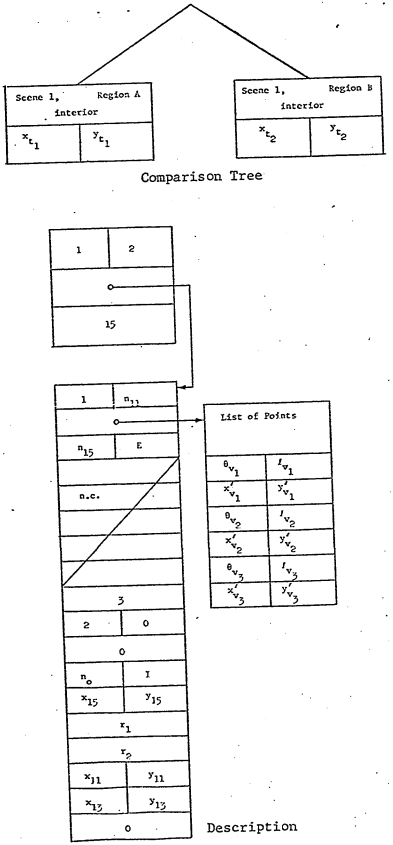


Fig. 3.17
100

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the two versions of R5 have the same center. At this point the comparison tree and description are as shown in Fig. 3.18, and the description of SCENE 2 is completed. Since there are two possible solutions, and because the relative rotation of the two scenes has not been determined, the process continues with the comparison of vertices in the order of their discovery. There are six of them, since each is described twice. Since each vertex is described twice, the intensity of the region associated with it each time is used to select the correct description, much as the "sense" was used before. When V1 is matched with its equivalent in SCENE 1, the relative rotation is determined, and the comparison tree is shown in Fig. 3.19. The "A" branch remains, for V1 could have been off the screen in SCENE 1. As the process continues, V2 and V3 are matched. Once the relative orientation of S1 and S2 is established, the identical nature of V2 and V3 poses no problem, since their expected location was known. On the "A" branch none of the vertices could find matches, since their spacing from Region A was different than the required spacing from Region 5. As long as there are two viable branches the process continues, and eventually all the vertices are matched on the "B" branch. The final comparison tree is shown in Fig. 3.20 and the "B" branch is selected because of the preponderance of matching elements.

The last example illustrated the desired performance of the program. Things do not always work so well, as the next example shows. Here, difficulties will arise when two dissimilar pictures contain one object in common. For example:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

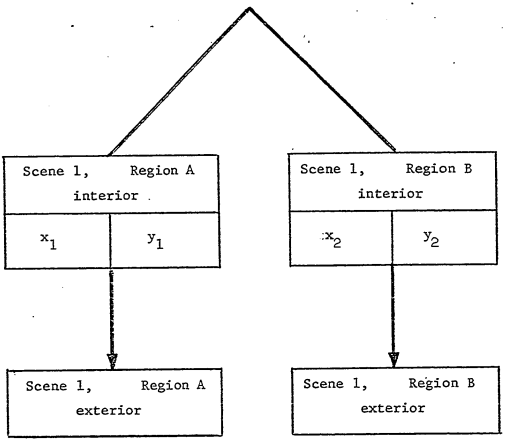


Fig. 3.18(a) -- Comparison Tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

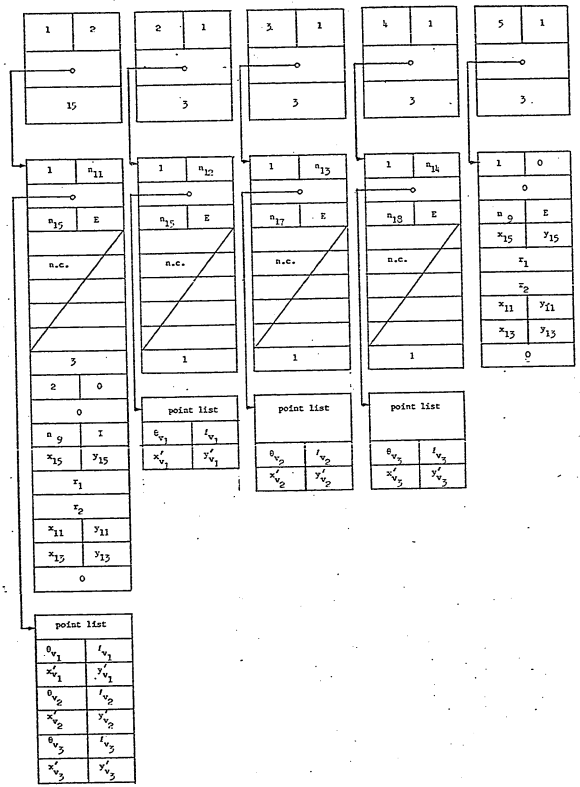


Fig. 3.18(b) - SCENE 2 Description

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

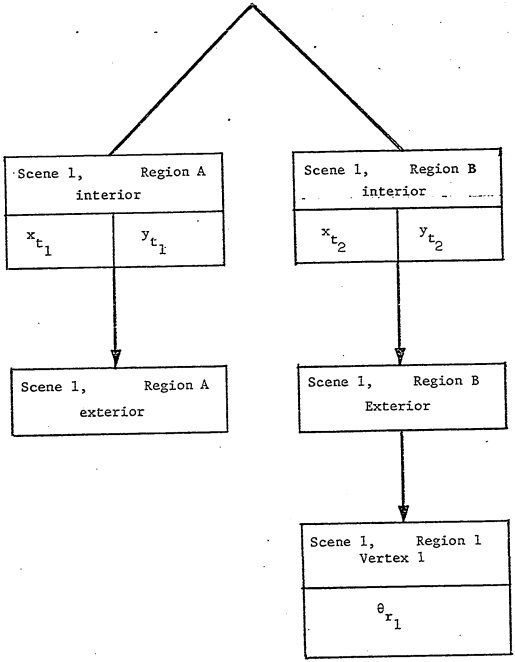


Fig. 3.19

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

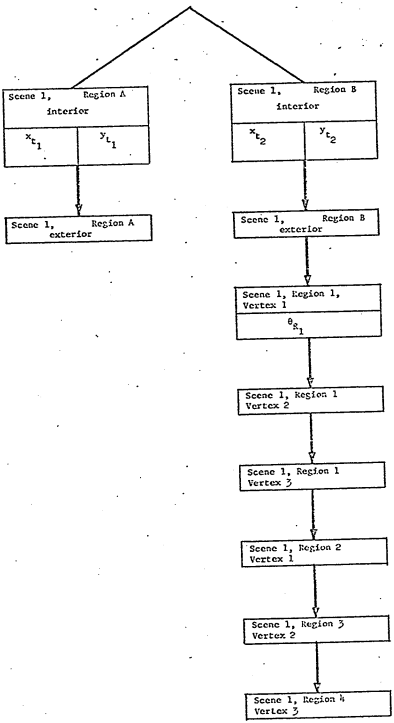
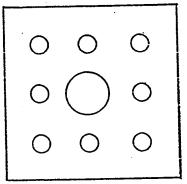
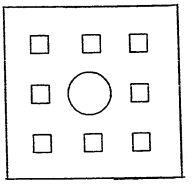


Fig. 3.20 - Final Comparison Tree

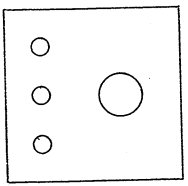
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



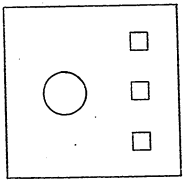
and



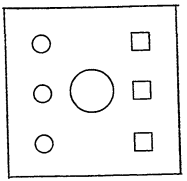
The big circles match, but the pictures obviously are not the same. The program is unable to distinguish this case from the following



and



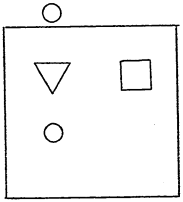
which can correspond to the real scene

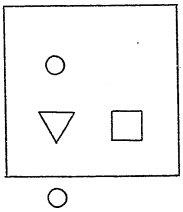


in which the images both represent views of the same scene. The program now rejects all matches of just one object, but at the price of muffing the second example.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Another failure mechanism can occur if the first object matched is improper and the correct match is off the edge of the screen. For example, two views of the same scene:





Here, if the small circle is matched to the only available circle in the second scene, the triangle and square will not match, and the program will say that the pictures are different. Only if the parse begins with the triangle and square will the comparison be successful, and this depends on the location of the objects in the scene. The problem is curable by trying the comparison several times with the objects in different orders, but the time penalty involved makes this method very unattractive. Fortunately, natural scenes do not ordinarily contain repetitive objects. Incidentally, this is why comparison trees are grown with the vertices at the end - since there is less information known about vertices than regions, the chances of a fatal misidentification are much greater.

This concludes the discussion of the operation of the picture processing system for the CART. In the next section some experimental results are given. Certain of the algorithms are discussed in more detail in Appendix II.

D. Experimental Results

This section is a compendium of experimental observations which I

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

made during the course of this study. Computer printouts of one particular picture identification problem are shown. However, the main focus of this section is a discussion of the relative difficulty of various subtasks within the area of picture description and identification, the computational cost of various subtasks, and certain difficulties of organization which I encountered which could be avoided in further work in this area.

Image processing is too large a subject for one person to adequately cover in a project such as this. In an attempt to get a system that worked and embodied the descriptive and description-matching features I wished to develop, I neglected or oversimplified several areas which turned out to play a key role in limiting the performance of the system. Among these areas are visual accommodation and operators.

Visual accommodation is the art of arranging the input hardware and pre-processing operations so that the data which one's program must analyze contains as much information and as little noise as possible. The length of a recent paper in this field(Tenenbaum [7])indicates the magnitude of the effort required here. An outdoor environment is subject to great changes in contrast and illumination, both as a function of time, and as a function of the vehicle location, and without effective accommodation neither this program nor any other could hope to be successful.

The scene conditions change so rapidly that the CART operator could not keep the input hardware correctly adjusted under manual control, long enough to get meaningful test runs. A great deal more is now known about computer visual accommodation, but this information became available near the termination of the research reported here, and is not incorporated into the programs.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The second area of neglect, that of "operators", results from my overestimation of the amount of progress that would be made in this area during the course of this study. Although in Section B, I discuss the advantages of region-operators over edge-operators, the actual operator incorporated in the program is sufficiently simple-minded to severely limit performance. I had expected to be able to develop the system using this simple operator and then switch to a better one at the end of the study, but a better operator with acceptable computational demands has not been developed. For this reason, I am restricted to pictures of very high signal-to-noise ratio, and cannot handle regions with uniform texture but irregular intensity.

Under these conditions, the absolute computational times taken by my program, and the reliability of its identifications are not useful numbers, based as they are on highly artificial examples. What is interesting are figures on the relative times required for various parts of the problem, and the relative amounts of program and data space required at various places. Section D1 presents this information, while Section D2 is a presentation of an actual analysis for the sake of completeness.

D1. Relative Computing Times, Data Spaces and Subroutine Sizes

The picture identification program, implemented in machine language and run on the PDP-10, requires between 3 and 5 seconds to construct a description of a picture and simultaneously match it with a pre-existing description. The overwhelming majority of this time is spent constructing the description, and only about 100 milliseconds is spent in description comparison. Once the description is built, successive matches take place very quickly. This is fortunate, because in order to overcome some of the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

difficulties mentioned in Section C, several matches using the description regions in different order may be required. For N objects, there are (2N)! different ways to do the matching (since each object has both an inside and outside outline which are treated separately). If a picture must be matched against several different scenes at once, the time involved could escalate dramatically.

The major reason for the wide difference between the description construction time and the matching time is the large amount of data which must be analyzed to construct the description. The input frame is roughly 256 by 256 and is in packed format, 9 4-bit bytes to a word. Thus on a 36-bit machine such as the PDP-10, the raw picture data takes almost 8k of core. In this format, the addressing of an intensity element is complicated in the x dimension by the necessity to compute both the displacement in words from the beginning of the buffer, and the byte position in the word. Even with byte manipulation hardware, the time is unreasonable. However, at one sample per word, the picture would occupy 64k words, and leave no room for any programs.

For this reason, my programs average the input data over a 3 x 3 grid and store the resultant 7-8k picture one sample to a word. Unfortunately, the byte manipulation hardware must still be used because the program needs working space to store the vectors used to outline regions. These are stored in other bits of the data buffer, so that the same address contains both the intensity of a picture element and the vector describing its connection to the remainder of the picture, as discussed earlier. More sophisticated operators might well compute more parameters per point than the one I use, and these also could fit in this word.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In this case, the buffer is small enough that I could have used one buffer for the intensity data and another for the vectors, and avoided using the byte manipulation instructions, which are among the slowest on the machine. This would have increased the program size, and although the CPU time per analysis would have gone down, the elapsed time might well have gone up, due to the preference of the time-sharing scheduler for small jobs.

By contrast, the number of words in a picture description varies from 100 to about 500, depending on the complexity of the picture and on whether the description has been compressed to eliminate unused words. Only compressed descriptions are stored for later use, but the description which is being made from the picture currently in core always has extra space in it for additional regions and vertices.

The disparity in size would become more intense if the vehicle was operated at a higher speed. In this event, areas further in front of the vehicle would have to be examined than is presently the case, in order to allow time for decision making and vehicle response time. Since the near field would have to be scanned as well, the lense would be wide angle, giving poor resolution of distant objects, and the additional loss caused by the averaging and reduction I now do would be unacceptable. In this event, the whole picture would have to be stored, resulting in a space requirement 9 times as large, and a description computing time nine times larger. In this event, some variant of Michael Kelly's scheme of "planning" would be useful, if one could figure out how to compensate for vehicle motion [8]. In the planning method, low resolution pictures are taken to isolate areas of significant interest, and high resolution shots of just these areas are taken to get detail. However, Dr. Kelly's application was to unmoving

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

pictures of faces, not moving road scenes. Thus the assembly of the high-resolution shots into a coherent whole was much easier than it would be here.

In spite of its large size, the picture data is a relatively simple structure. This is born out by the relative sizes of the sub-routines involved in the analysis and description process. The basic routine to reduce the input data is only about 100_8 words long. The routine to create the vectors which divide the picture into regions is about 1000_8 words, while the routine to make straight line approximations to outlines and identify vertices is about 700_8 words long. Even including the routines which do the data management of the picture description (insert regions, add vertex descriptions, etc.) the total length is under 3000_8 words. The buffer which these routines manipulate is almost three times as large as the routines themselves. By contrast, the routines for description comparison amount to 3400_8 words, while the data structures involved here are typically only 200_8 to 500_8 words long. The difference, of course, springs from the fact that the processing on the actual picture is very repetitious, but quite simple. The picture description is a more complex structure, but much shorter (which was the whole point of creating it). Thus the code to manipulate descriptions is only executed a few times, but must possess a great degree of generality to handle the wide variety of possible descriptions.

The major failing of my program from the organizational standpoint is one that it shares with every other such program I am aware of. The amount of descriptive information which it gives about a picture is highly variable. In effect, what the operator does is to specify a level of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

detail "d" at which the picture is to be described. After the parameters (fudge factors) which determine "d" are set, all pictures are analyzed the same way. If the picture has lots of structure, the level of description complexity "p" is very high. If the picture is very constant over much of its area, "p" will be very small. Since the pictures encountered by an outdoor vehicle vary widely, what is really wanted is a way to set "p", so that all pictures are described to the same level of complexity. In a very complex picture, only the most important or prominent things are described; in a simple picture, finer detail is included. If "p" is a measure of the number of descriptive elements in the picure, it is obvious that the confidence level of a match between two pictures is an increasing function of p. So is the time to compute the match. If p cannot be set directly, but must be controlled through d, we run into difficulty.

If d, the level of detail, is set high, in order to ensure a sufficient p for simple pictures, then for complex pictures p will be large and the matching costly. If d is reduced, then for simple pictures p will be too small to sufficiently characterize the pictures.

Speaking less abstractly, if one is too picky about what constitutes a homogeneous region, one can divide up a physical object into many regions of no physical significance based on variations of shading and surface texture. If one isn't picky enough, one will fail to separate genuinely different objects. By the same token, one does not wish to separately describe each leaf on a picture of a tree, even though they are actually separate. The program really needs to set the level of detail "d" so as to best describe the major picture features and leave out the trivia.

This whole area is an extension of accommodation from the input

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

hardware to the analysis program, and another application of the concept of feedback to computer programs. Ideally, a program would keep track of the value of "p", and iteratively change "d" until p was acceptable. Unfortunately, in the present program the delay time around the loop is too long for this method to work, due to changes in the input scene. Further, if the same scene is saved and reanalyzed until success, the convergence time is too long for real-time operation.

Let me summarize my views at the conclusion of this experimental study as follows: Although the overall structure of my approach to picture identification is valid and useful, the specific implementation of it for automobile control is not. The computing time requirement is too large, and the "picture resolution/core storage requirement" is too large. The reliability is unacceptably low, and inclusion of more reliable accommodation and homogeneity operators to increase reliability would increase the computing time by an order of magnitude or more. I regret to say that I can offer no specific suggestions to remedy this situation.

In the field of remote planetary exploration both the processing time and low reliability could be tolerated. A further advantage of this application is that the high cost of the equipment (over $250,000) required is not a major factor in the overall cost, whereas for the automobile application the cost is a prime factor. Again though, direct application of my experiments will be difficult, since our experimental setup did not permit the investigation of the off-the-road control situations.

D2. Typical Picture Identification by Computer

In this subsection, we will consider the description and identification

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

of a simple picture, consisting of a triangle and an oblong on a dark background. I have previously exposed the program to a different view of this picture, and have stored the resulting description on off-line storage.

I present the analysis as a commented dialog between the computer program and myself, referring to various figures which indicate the state of the internal processing at various times. My statements are prefaced by "*", program statements are prefaced by "#" and the commentary in unprefaced.

*RUN RGROW

#OLD FILE NAMES

*SCENE2

Here I instruct the program to read in the stored description of the scene "SCENE2". A drawing of this scene is given in Fig. 3.22. This scene is keyholed because of perspective transformation. The input picture was rectangular, but the actual scene was lying flat on the table, and the camera was looking down from an angle. Thus the bottom of the scene showed less width than the top. The description from which this drawing was made is contained in Fig. 3.23.

#INPUT FILE NAME

*SCENE

Here I direct the program to read in the actual picture data file "SCENE". Fig. 3.21 is the drawing of this scene. This is an unprocessed direct copy of a TV image. If I had said *TV I could have gotten an actual TV picture instead, but I use saved TV images for repeatability of experimental results. The data read in is printed (after reduction) in hexadecimal

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

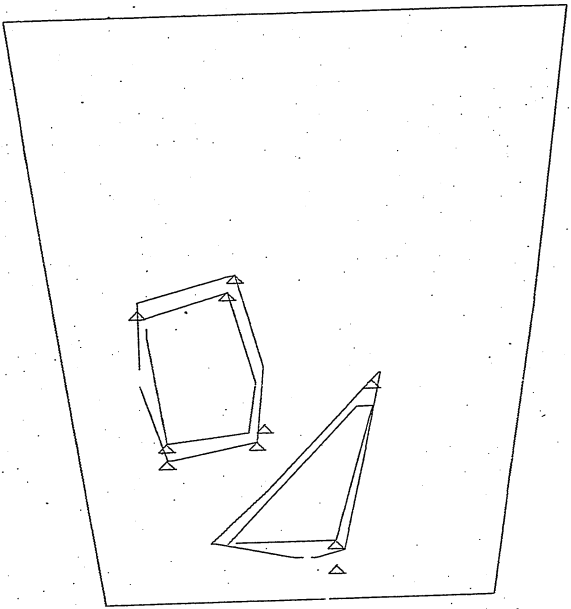


Fig. 3.21 - SCENE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

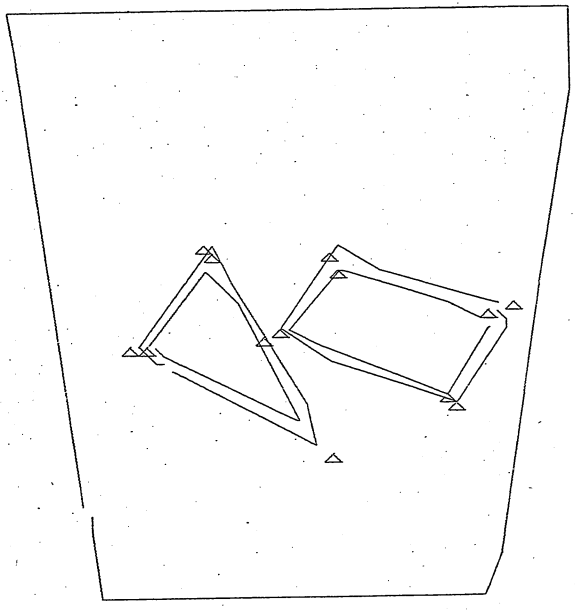


Fig. 3.22 - SCENE 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
52772/        ,  1436  52773/     110,    110  52774/        4, 52775  52775/        1,      3
52776/      15, 51353  52777/  -     1,      1  53000/        2,      1  53001/       15, 51463
53002/        ,    16  53003/       3,      1  53004/       15, 51514  53005/        ,     16
53006/       4,        53007/      15, 51540  53010/        ,     10  53011/        1,     15
53012/      15, 51406  53013/  -     1,    424  53014/       50,     15  53015/  204554,-111521
53016/  203530,-262727  53017/       5,      2  53020/        7,-     11  53021/        ,
53022/       2,    11  53023/      15, 51423  53024/        ,    111  53025/       57,     47
53026/  204701,110314  53027/  203565,134533  53030/        3,-      5  53031/  -     16,      1
53032/        ,     4  53033/       3,     11  53034/       15, 51444  53035/        ,    100
53036/      34,    53  53037/  204712,127135  53040/  203417,-361115  53041/        3,-      3
53042/  -     3,-    16  53043/        ,      3  53044/       11,     74  53045/        2,      7
53046/       1,     1  53047/     104,      1  53050/      104,     13  53051/      100,     42
53052/      76,    57  53053/      73,    102  53054/       71,    107  53055/       13,    107
53056/      12,    77  53057/      12,     76  53060/       12,     75  53061/       73,     44
53062/      55,    40  53063/      50,     35  53064/       41,     47  53065/       47,     53
53066/      66,    60  53067/      74,     47  53070/       74,     46  53071/       73,     45
53072/  201475,-367553  53073/      47,     36  53074/  201734,260225  53075/       37,     50
53076/  201515,-307613  53077/      66,     60  53100/  201741,- 57613  53101/       75,     44
53102/      24,    54  53103/      45,     65  53104/       44,     60  53105/       33,     41
53106/      31,    35  53107/      20,     51  53110/       21,     52  53111/       22,     53
53112/      23,    53  53113/  202514,-107316  53114/       47,     66  53115/  201765,242341
53116/      30,    35  53117/  201672,250341  53120/       17,     51  53121/        1,     10
53122/      15, 51474  53123/  -     1,    102  53124/       57,     47  53125/  204641,165163
53126/  203436,156747  53127/       2,-      4  53130/  -     15,      1  53131/        ,      4
53132/      72,    47  53133/      65,     57  53134/       46,     51  53135/       42,     47
53136/      50,    40  53137/      65,     44  53140/       71,     46  53141/       72,     46
53142/  201544,160131  53143/      65,     57  53144/  201746,-357653  53145/       41,     47
53146/  201451,-237653  53147/      50,     40  53150/  201724,-207647  53151/       72,     45
53152/       1,     7  53153/      15, 51525  53154/  -     1,     67  53155/       33,     52
53156/  204601,251371  53157/  202624,-164761  53160/        3,-      1  53161/       11,     10
53162/        ,     2  53163/      21,     51  53164/       30,     40  53165/       34,     44
53166/      43,    62  53167/      25,     53  53170/       23,     52  53171/       22,     51
53172/  202402,-157576  53173/      31,     36  53174/  201704,-357576  53175/       21,     51
```

Fig. 3.23 - SCENE 2 Description

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

in Fig. 3.24.

#COMPARE?

*YES

The program needs to know whether to merely create a description for SCENE or to simultaneously compare it with SCENE2.

(delay of 5-15 seconds, depending on machine workload)

#DONE

At this point, the description of SCENE is complete, and the tree structure describing the correspondence between SCENE2 and SCENE is fully developed. The description is shown in Fig. 3.26 and the matching tree in Fig. 3.27. Note the excess space in the description of SCENE. This would be eliminated if I requested that this description be saved.

*TIME

#3.200

The time to do the analysis was 3.2 seconds of CPU time.

*PARSE

At this point I ask the program to display the correspondence between the two pictures, and the resultant superimposed display is shown in Fig. 3.28.

The remainder of this section is a guide to the interpretation of the various figures already mentioned.

Fig. 3.22 - SCENE2

Two boundaries for each object are shown, the boundary of the dark background and the boundary of the light object. The small triangles mark the location of the vertices found by the program. Note that not all the vertices are found, and that some do not fall on the boundaries of

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

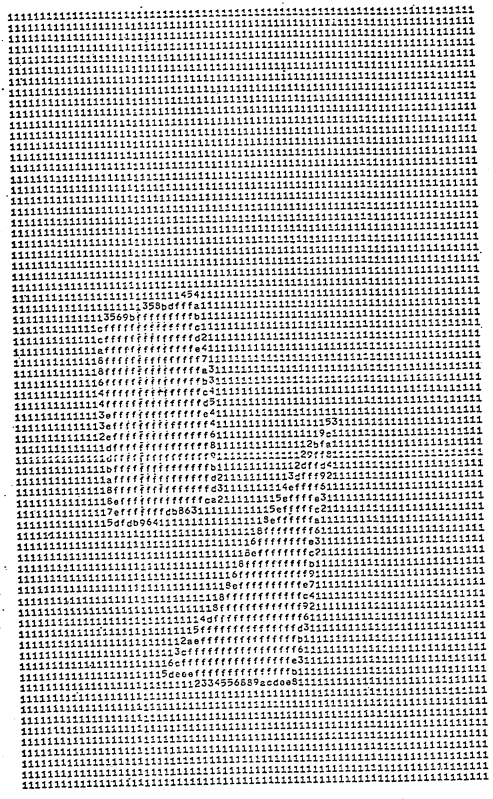


Fig. 3.24 - SCENE Input Brightness

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

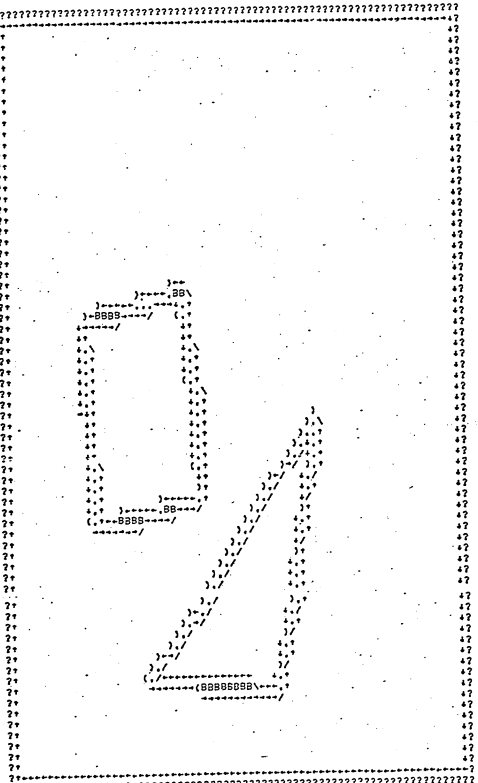


Fig. 3.25 - SCENE Region Map

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

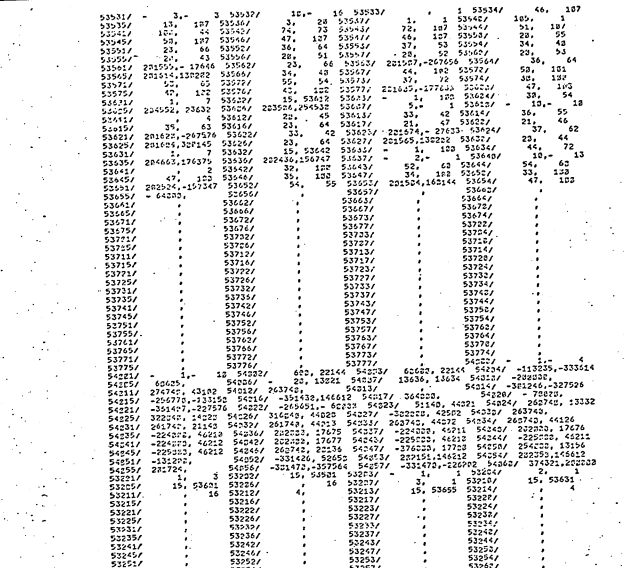


Fig. 3.26 - SCENE Description

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

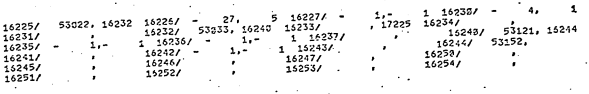


Fig. 3.27 - Comparison Tree

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

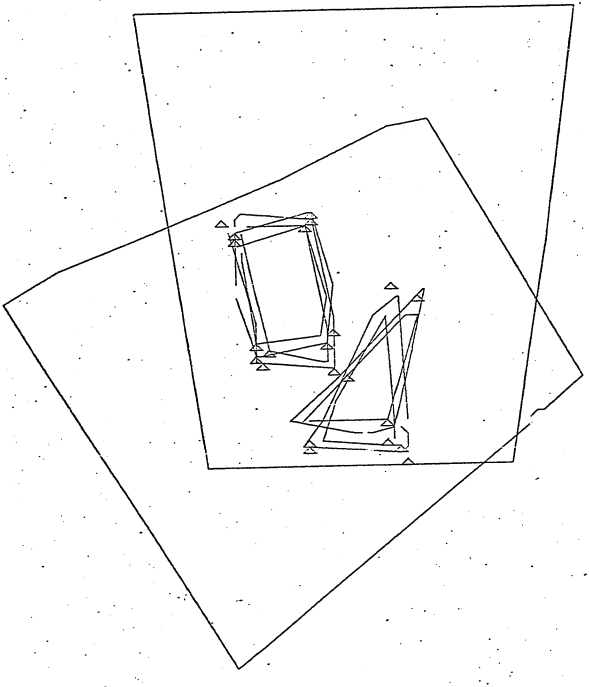


Fig. 3.28 - Final Picture Matching

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the object in question. The two reasons for this are roundoff errors in the computation, and the computation of vertex locations from extensions of lines rather than from the literal meeting of lines. This second effect is necessitated by the "knocking off" of corners which sometimes occurs in the line-fitting process. This is illustrated by the extreme right hand corner of the oblong in Fig. 3.22 and the lower corner of the triangle in the same figure. The vertex location is computed from the extension of nearby lines.

Vertex location is an imprecise business at best, for the line-fitting algorithms are not exact. If the goodness-of-fit requirement is set too high, many lines will be required to outline even the simplest object, due to the irregularity of the input data (see Fig. 3.24). Thus the line structure in the description will bear no geometrical relation to the true structure of the object. On the other hand, too sloppy a requirement leads to corner-wrapping and neglect of very obtuse vertices. Since the time to fit lines increases drastically with the goodness-of-fit, the requirement was set relatively low here.*

As mentioned earlier, the "keyhole" effect is due to perspective. In the drawing, the viewpoint is from directly above, while the picture was taken from an angle.

*Although the algorithm for best fit to a set of points is computationally simple, the problem here is one of partitioning a set of points into an unknown number of subsets containing an unknown number of points to which lines will be fit. This can require many iterations of tentative partitions until the right one is found.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Fig. 3.23 - description of SCENE2

This is the description from which the drawing of Fig. 3.22 was made. The general outline of a description has already been given in Section C, and will not be repeated here. Only the new features not mentioned previously will be specifically mentioned.

First in the description is the header information required to interpret this description. Since the description is heavily laced with pointers, but was created in one area of the program (the area now being used by SCENE) and is now stored in another, an offset must be provided to these pointers. The value in cell 52772_8 is this offset - 14368. Cell 52773_8 is the size of the picture described here - 110_8 x 110_8. The next cell contains the number of regions in this description - 4, and the actual starting location of the description - 52775_8. The next 12_{10} locations - up to 53010 - are the headers for each region. Region four is a dummy region with no information, - a vestige of the picture description algorithm. The number "15" located in the left half of cells 52776, 53001 and 53004 indicated that the displacement which is to be added to the right half of these cells to get the true address is in index register 15. In order to use this description, the value 14368 must have been placed there. The PDP-10 address computation hardware will do the addition automatically. All the other pointers in the description are similarly indexed.

The remainder of the description is the sets of parameters characterizing each region, together with the locations of the vertices and the locations of the points defining the outlines drawn in Fig. 3.22.

Fig. 3.24 - intensity map of SCENE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This is a hexadecimal printout of the brightness of the input picture. Note that the boundaries of areas are not sharp and that lines which are not parallel to the grid structure of the picture are "jaggedized". This is where spurious vertices can easily be introduced.

Fig. 3.25 - region map of SCENE

Here the intensity map of Fig. 3.24 has been broken down into three regions. The conventions in the printout are that arrows are vectors in the obvious directions, /'s are up-and-to-the right arrows, backslashes are up-and-to-the-left,)'s are down-and-to-the-left, and ('s are down-and-to-the-right. Dots and B's are various kinds of isolated points. Blanks are interior points and ?'s mark the outline of the picture.

In this picture one can see that the points of intermediate intensity along region boundaries are not included in either region, further "jaggedizing" the regions. There is no simple way to decide whether or not to include these points in the general case.

Fig. 3.26 - description of SCENE

This is similar in structure to Fig. 3.23 except that the header information is not stored with the description, and that the offset for the pointers is zero. Also, there is a lot of extra space in the description in case the input picture had required a more complex description.

Fig. 3.27 - correspondence tree between SCENE2 and SCENE

The tree begins at location 16225_8. According to the tree, the first region outline in SCENE corresponds to the outline beginning at location 53022_8 in the description of SCENE2. The displacement between the two outlines is -27_8 in x and 5 in y. The contents of locations 16227 and 16230 indicate that the program initially found a second match, but

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

subsequently discarded it because of some contradiction. Since cell 16225 points at cell 16232, the correspondence map continues there. This cell says that the next outline in SCENE is matched by the outline beginning at 53033 in SCENE2. The cell following says that the rotation of the two pictures relative to each other is contained in cell 17255, which is not shown here. Since cell 16232 points to cell 16240, the tree continues there. This cell says that the third outline in SCENE matches the outline in location 53221, and that the tree continues at location 16244. This final cell says that the fourth outline in SCENE matches the outline starting in location 53152 or SCENE2, and that the tree ends here.

In this case, no vertices were compared, and only a single branch of the tree remained at the end. If there were multiple branches, the vertices would have been used to try to eliminate all but one of them, and the vertex comparison data would have appeared at the ends of the branches. The vertices are done last because of the previously-mentioned difficulties in correctly determining them.

Fig. 3.28 - final result

This picture is the final output of the program, describing the correspondence between the two scenes. In an operational program, this step would be eliminated, for the decision making algorithm (described in the next section) needs to know only <u>that</u> two scenes match, and not the details of <u>how</u> they match.

It has been my intention in this section to present the program as an example of what can be done, rather than as a finished product. I think the approach is conceptually correct, but that the details depend heavily upon the application. My choice was to restrict the program to fit the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

computation time available to a real-time application. The result was a program insufficiently reliable to use. The alternate approach would have been to write a more reliable program which took longer, but such a program could not be used in a real-time environment. If this alternate approach is pursued, it should be done with the knowledge that the product will not be useful in the area of automatic automobiles, but perhaps in the area of remote exploration. A more likely area, given the currently declining support of planetary exploration, is the automatic control of sophisticated manipulators in heavy industry.

E. Suggestions for Automization of the Navigational Decision Process

The remainder of this chapter is a discussion of the way that this picture recognition capability could be used in a system of vehicle navigation. The mechanism is basically a combination of map-reading, dead reckoning and the visual recognition already described. This part of the process (maps and dead reckoning) has not been programmed at the A.I. Project since the versatility of the perceptual system does not warrant it. The robot work at SRI ("Shakey" et al) indicates the type of thing that can be done.

The first prerequisite for a journey is a route from the start to the destination. Although one might reasonably leave the job of selecting the route to the human operator of the vehicle, it is not too difficult to automatize this process. If one regards a road system as a maze, rather conventional game-playing techniques can be applied to the problem. This area was not explored as part of the CART system, primarily because the sorts of "mazes" required are so trivial that no worthwhile research

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

contribution would result.

Given a method for finding a route, the next problem is to specify the route information in such a way that a vehicle travelling along the route can determine which intersections are upcoming and be prepared to recognize them when it gets to them. Allied problems are figuring out how to tell the vehicle what action to take at the upcoming intersection, and developing an implementation which does not require descriptions of ALL the intersections along a route, but only those at which action is required. This is virtually mandatory if the vehicle is to be used for extended journeys where hundreds of intersections may be encountered.

Were I to add such a feature to the existing programs, the organization would be a set of interconnected nodes, where each node represents a perceptually identified location, and the edges between nodes describe permissible paths between nodes. The edges would be characterized by their length, and the nodes by the number and angular spacing of the edges emanating from them. For the trip from my home to work, the map would be:

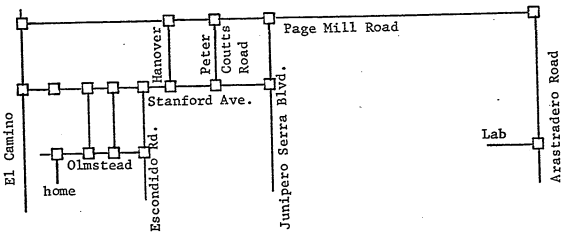


Fig. 3.29

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and a sample node would be:

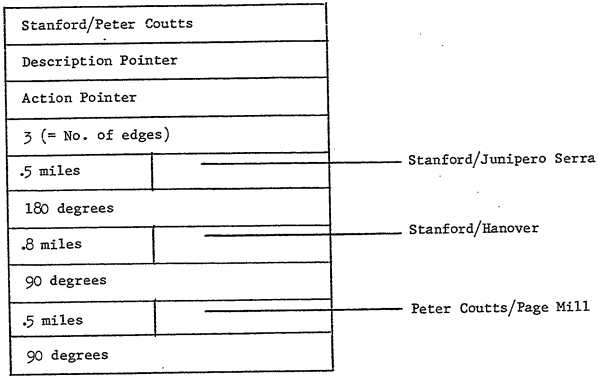


Fig. 3.30

The description pointer would point to the perceptual description of the intersection in question, while the action pointer would be preloaded for a particular journey to indicate what action should be taken upon getting to that intersection (depending on how I wanted to get to work that morning). As the journey progressed, the map would indicate the identity of the next intersection, and the dead reckoner would compute the expected arrival time. This way, only nodes that are relevant to the journey need be included in the map, since intersections along the route but far removed from the desired one will be ignored. Any intersections (driveways) near "Stanford/Peter Coutts" will be checked, but they don't look like the intersection, so my map need not even include them.

The dead reckoner required for such a system would not need to be anymore accurate than a road map or vehicle odometer. This is particularly

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

desirable when one considers the human/vehicle interface. If I have to specify the distance to my destination to the nearest foot, my enthusiam for my automated automobile will be somewhat damped. On the other hand, if "Oh, about fives miles" is good enough, I will find my vehicle a useful and timesaving tool. Of course, there is no difficulty in measuring the distance during the first trip so that succeeding trips become even easier.

Let me emphasize that I do not consider the area of navigational decision making to be a serious problem area, even in the case of remote exploration vehicles. The difficulty is in the perceptual area, and it consists of obtaining and correctly analyzing sufficient data to enable the system to determine the vehicle's location. Without location information, no reasonable navigational decision can be made, and with location information, no reasonable navigational decision is difficulty. I wish to separate out the class of decision which is an attempt to determine whether some route is physically possible for the vehicle to take. This is not part of navigation, as I construe it, and except for obstacle avoidance of the most immediate sort, can be handled by remote guidance for exploring vehicles.

This concludes the discussion of navigation. In the next chapter, I will discuss incident avoidance, conceptual description, and intensity resolution - problems which are not currently soluble for a vehicle moving in an actual road environment, and which in my view are not likely to be solved any time soon.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER IV

OBSTACLES TO FUTURE WORK

The intent of the research described in Chapters II and III was to examine critical features of guidance and navigation in the automated automobile context. During the course of this research, experiments were performed in both guidance and navigation using the experimental system. A third ingredient necessary in an automated vehicle system is a mechanism to avoid unforeseen incidents, or accidents. Such a mechanism was not developed. This was only partly due to time and money limitations. The author believes that such a mechanism is, for the automobile case, beyond solution by extrapolations or extensions of any hardware or techniques currently available.

The purpose of this chapter is to discuss several problem areas which are limiting to automobile automation as considered here. These areas can be sidestepped by suitable task structuring in applications areas such as remote exploration or industrial automation. However, they cannot be avoided in automobile control. The content of this chapter is conjecture. However, it is conjecture backed by the research described elsewhere in this document.

Section A below discusses the problem of image intensity resolution. This problem is the least serious of those discussed, since it is likely that it will eventually be eliminated by advances in electro-optic technology. Section B discusses scene description, which poses problems not subject to technical breakthroughs. The construction of appropriate

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

descriptions of real world, three dimensional scenes requires more information than the content of the images. Required techniques therefore necessarily go beyond the present state of artificial intelligence work.

Section C describes the problems involved with extending conceptual processing to anomalous situations (unforeseen incidents or accidents). These situations, in addition to being unexpected, may involve extremely subtle inferences. An example of such a situation, as handled by a human operator, is as follows:

> I am driving down the road on a moderately windy day. I round a curve and sitting on the road right in front of me is a large box. I must decide whether to make a violent evasive maneuver or simply run over the box. In order to make this decision I must decide whether there is anything in the box. If it is empty, it is surely stupid to take the chance of wrecking my car to avoid it. On the other hand, if it is full of sacks of cement, I will surely wreck my car if I hit it. It might even have a child hiding in it, depending on the circumstances. One key item of information which I can use to judge the situation is the effect of the wind on the box. If the box is firmly rooted to the pavement while everything else is being blown by the wind, the chances are there is something in the box. If the box is blowing with the wind, it is probably empty. If the box is rocking violently, but the wind is not that strong, there may well be someone in the box. I make such a decision instantly, with no special effort.

Section C attempts to point out that incident avoidance would require computer capability which would indeed approximate that of the human brain.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

With this introduction, let us proceed to the main body of the chapter and discuss each area in turn.

A. Dynamic Range of Illumination in Outdoor Scenes

Existing photoelectronic sensors (vidicons, image dissectors, etc.) are capable of reproducing a dynamic intensity range of about 60:1 within a given picture. The range of illumination in an outdoor scene can easily exceed 500:1, including specular reflections and deep shadow. The human eye can accommodate this range without damage or serious loss of information, but in a photosensor, the bright portions produce "halos", "comet-tails" or "burn-in", while the dark areas blend into featureless obscurity. If permanent sensor damage can be avoided somehow, this problem can be alleviated somewhat for stationary scenes by taking multiple images at different sensitivity levels, but for vehicle applications the motion of the vehicle makes this approach impossible. This motion also rules out such photosensors as the image dissector, which is a random access sensor with very high spatial and intensity resolution, but which takes hundreds of seconds to input an entire picture.

Fortunately, intensity resolution problems do not preclude all visually guided vehicles. Indoor vehicles operate under far more uniform lighting conditions, and remote exploration vehicles need not be constrained to process images taken while in motion. It is only the "Automatic automobile" concept which this problem affects seriously.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

B. Stereo and Depth Perception, 3-D Description

A different sort of processing difficulty arises in scenes which contain 3-D objects scattered over a large area. Many road scenes are like this. Consider the road to my lab, with trees along the right side, a hill on the horizon, and a house on a corner on the left. Notice that in 18 words I have given a description adequate for a human driver to identify an intersection. It is a conceptual description, however, containing no measurement of the height of the hill, the color or number of windows in the house, or what species the trees are. Pity the poor computer trying to describe all this from an essentially 2-D picture in such a way that it will arrive at an equivalent description when travelling the other direction on the road, and seeing the other side of the house. The traditional solution for this difficulty is to call for steroscopic vision and a 3-D computer description. But this is of little help, for if the house is 100 meters away, the stereo cameras are 1 meter apart (pretty far for a car) and the lens is the usual 12.5 mm wide angle lens, then the displacement of a point on the image planes of the two cameras is .125 mm (about the same distance as the picture element spacing). Thus stereo range finding will be very inaccurate, and hardly suitable for a good description. A human gets around this problem by remembering the conceptual unit "house" rather than a mental picture of exactly what the house looks like. Even if his purpose requires a mental image, the image is enhanced by his prior knowledge of houses (e.g. they have vertical walls, and are generally block-like in outline) which enables him to make much better use of the perceptual image he has. The reliance that a person places on such information is nowhere made clearer than in the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

optical illusion produced by a room with slanted walls and ceiling which makes its occupants appear as giants or midgets to each other. The conceptual identification of the room makes this illusion possible, but in normal circumstances it makes human vision a far more powerful tool than computer vision without conceptual back-up.

Again though, the requirement for conceptual description arises only in the case of an automatic car. For an exploration vehicle on another planet, the variety of objects would be small - mostly rocks and terrain features, of which a conceptual description would be of no value. For an industrial vehicle, the objects in its "universe" could be controlled, and if necessary marked in a way which would guarantee their uniqueness without affecting their usability to human workers. Only the automatic car must face a varigated world in which few special provisions can be made for it.

C. Incident Avoidance, Anticipation, Use of Experience in Decision Making

The foregoing arguments do not demonstrate the infeasibility of an automatic automobile - they merely argue in favor of some other approach than computer vision. The area of incident avoidance is the crucial area in which vision - or an equivalent imaging system - is required. It also requires almost all of the "common sense" knowledge that a man has about the world, along with most of the intellective methods that his mind is capable of. Let me present these arguments in several sections. First I will discuss the nature of "incident avoidance" including the possibility of designing the problem out of the system. Next I will demonstrate the requirement for an imaging sensory system, and lastly I

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

will argue (but not "prove" in any mathematical sense) that the knowledge required to operate upon the image data is effectively unbounded, and hence requires the computer system to possess the same intellective mechanisms and a fair fraction of the same knowledge as a human, in order to develop the same driving capability as a human.

To begin with, an "incident" is a situation requiring action which was not anticipatable at the time the journey was planned. It includes everything from merging between two other cars to avoiding children running in the street. For some classes of "incidents" such as merging, a general solution to the problem can be worked out in advance and implemented appropriately when required. Hardware systems have been produced which sequence cars onto limited access freeways by sensing other cars and varying speeds appropriately. In this case, the necessity of invoking the merging program or hardware is easy for the vehicle controller to detect. Other classes of incidents are not so obvious. Suppose we are driving down a city street next to a line of parked cars. What is the cue that the driver of a particular parked auto gives us that reassures us that he is not going to open his door and step out in front of us? Lacking this cue, what should we do? Honk, change lanes, slow down, panic stop, pray, or what? The answer depends on our analysis of all the factors of the situation and our prior knowledge about them. Factors such as lane width, traffic density and the age of the other driver all enter into such a decision. Thus "incidents" can be simple algorithmically specified situations or complex problems involving prior knowledge, correct selective focussing ("heuristics") based on experience, and ethical judgement (To what length should I go to avoid running over a man's wallet....

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

his dog...his child?)

These problems are inherent in any system of automobile transportation because of its unique requirement of sharing space with other vehicles and objects. The thing that distinguishes automobile and truck transportation from other forms is that autos and trucks take their cargo all the way from its source to its destination without signficant use of other methods of transportation - at least this is the intent of their users. Other forms of transportation under somewhat controlled conditions. The terminals are relatively few in number (thus allowing them to be expensive) and are often located some distance from the ultimate destination. This fact, along with the nuisance of scheduling present in other systems, is what induces most people to use automobiles, in spite of parking problems, congestion, delay and the host of other problems involved. A marketable automatic car must therefore be prepared to offer this same service. John McCarthy has suggested that a typical use for such a vehicle might be to take its owner downtown for an appointment, drop him off at the correct address, and go away until needed - either returning home until the appointment is concluded, or finding a parking space in an outlying area. Clearly then, the destination area is going to be full of pedestrians, as well as the ordinary artifacts of commercial or residential living - garbage cans, packing crates and the like, some of which may interact with the vehicle. Of course one can envision a science-fiction type city in which all these problems are avoided, but in the context of today's urban centers they are ever-present.

The non-computerized automated cars which have been considered by

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

other authors have gotten around these problems by postulating operation only on limited access roads. As we have seen, there is little economic value in this if a driver is still required at both ends - but even this is an oversimplification. No highway can be truly "limited access" if anyone is using it at all. First, there will always be objects dropped from vehicles - either parts of the vehicle such as hub caps, tire tread, or mufflers - or part of the load, such as boxes, newspapers, or even stepladders (it happened to me once!). Secondly, if conventional cars are allowed, there will be drunk, drugged, or sleepy drivers, whose actions cannot be anticipated, nor necessarily avoided in any mechanistic fashion in heavy traffic. A human driver, observing erratic behavior or an improperly secured load, can give such a hazard a wide berth, but a hardware driving system would have a hard time detecting either hazard.

Having shown that "incidents" are inherent in driving, let me proceed to show that an imaging sensor is necessary to deal with them. No argument here should be construed to show that this is sufficient, however. The definition of an "imaging sensor" is a contextual one. Basically, an imaging sensor must provide information about a scene as it exists at a particular moment in time, with spatial and intensity resolution sufficient to separate objects relevent to the purpose at hand. Thus, for a stationary scene, an image dissector is an imaging sensor, while for a moving scene it is not. A microwave radar is an imaging sensor for the purpose of locating planes, but not for the purpose of identifying them. Simply speaking, an imaging sensor provides information telling WHAT something is; a non-imaging sensor tells only THAT it is, and maybe WHERE it is.

By this point, it should be obvious that in the driving context

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

discussed earlier, an imaging sensor and an image processing program are required. It is not sufficient to know that there is an object to the right and in front of one's car - one must also be able to tell that the object is a parked car, that it is occupied, and that the driver appears to be reaching for the door handle without checking for oncoming traffic. It may well be that this is beyond the capacity of the image processing program (which is why my preamble contained no argument about the sufficiency of imaging sensors) but it is certainly impossible without the information which is contained in the image.

Let me conclude with my most controversial argument - namely that incident avoidance, which subsumes all of the decisions required to avoid causing accidents, anticipate and avoid hazardous situations, and minimize the consequences of unavoidable emergency situations, is an open ended problem requiring not only most of a man's "common sense" knowledge about the world, but also the capacity for conceptual reasoning, hypothesis information and verification (learning), value judgements and ethical decisions. In short, the process of driving, in its fullest context, is of a fundamentally different sort than the process of algorithmic problem solving. A man, having learned calculus, may embody his knowledge in a computer program which may then solve calculus problems faster than he can. But he cannot in the same way write a program embodying his knowledge of driving, for this is not a static set of rules which he can transliterate into ALGOL. Driving a car has more in common with LEARNING calculus, and learning is a meta-process which a person cannot describe in the same manner as the subject matter which is learned. [Learning, as used here, is the process of abstracting information from reality and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

incorporating it in a conceptual framework so that it may be referenced for decision making, further learning, or prediction. It is not synonomous with any and every form of self-modifying behavior.

If by some chance a program to do this (learn) were created, it would be as capable of learning calculus as driving, as capable of learning any one human skill as any other. In fact, such a program would possess all the intellectual potential of an uneducated human being. To those who feel that there is no difference in principle between a computer and a human being, this argument is no doubt unconvincing. But to those of a more engineering orientation, who are interested in automated vehicles as a useful transportation device, or who pursue the goal of a computer-driven auto thinking that it is surely simpler than building one of Dr. Asimov's robots, I hope that they see that they are mistaken. If safer, more convenient transportation is the goal, there are ways to achieve it, but the only justification for the insertion of artificial intelligence into the search is an expectation of a computerized vehicle compatible with the human controlled vehicles and environment we have now, and this expectation is groundless.

Even the most fervent advocates of the functional equivalence of man and computer do not deny the vast difference between the mode and size of memory and processing abilities of man and computer. As far as we can tell, the fundamental units in a man's mind are images, symbols, and concepts, linked associationally and hierarchically, where a computer deals with such things as aggregates of bits possessing no especially nice qualities from the processing, retrival or data management points of view. Barring an utter revolution in computation technology, future research in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

computer control would be more profitable in areas such as industrial automation, remote exploration, or man-machine systems, where either the problem can be algorithmically or heuristically bounded, or the mission permits interaction with human guidance to gain the benefit of computer reliability, speed, and efficiency on computationally simple portions of the problem, while occasional decisions involving high-level judgement are handled by a human supervisor.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

APPENDIX I

HARDWARE DETAILS OF THE CART SYSTEM

The CART project hardware came from many sources. The vehicle itself came from the Stanford Department of Mechanical Engineering; the computer system and TV interface was already in use at the Stanford Artificial Intelligence Project by the Hand-Eye project and others. Much of the control and interface equipment was designed and built by the author. An overall block diagram of the CART system is shown in Fig. A.1.

The PDP-6 and PDP-10 are used for a wide variety of time-shared A.I. research activities. Many of these center around the analysis of digitized TV images. These images are read from a conventional TV camera by a specially designed A/D converter built by William Wichman (see reference in Chapter 1), and transmitted directly into a main core storage by a DEC Type 167 Data Channel, which is also used for a high-speed swapping disk. The TV data is taken in the form of a parallel 4-bit grey-coded samples, packed 9 to a computer word, at a sampling interval of about 155 ns.

When I arrived at the A.I. Project, the TV A/D was connected directly to the TV camera used by the Hand-Eye project that Wichman had worked with. In order to increase the flexibility of the system I designed and built an 8-way camera multiplexer which allowed system users to computer-select which camera they wished to read. This allowed the CART project and the Hand-Eye project to time-share the A/D converter. Included in the multiplexer is a cursor injection feature which brightens the area on the TV monitor which is actually being read into core. The CART camera was

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

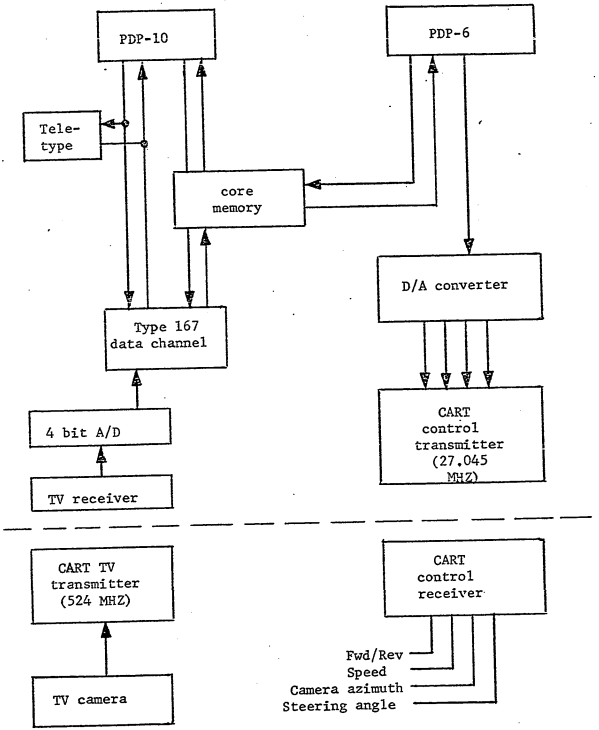


Fig. A.1 - Block diagram of CART system hardware

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the first camera to be connected with the system which was not direct wired. This required modifications to the A/D converter to enable it to decode sync pulses from the camera without a stable d.c. level of the input signal. This necessitated a floating sync pickoff threshold, which the author incorporated into the design.

The TV receiver was adapted from a scrapped UHF TV receiver. The only changes (beyond repackaging) were to provide slower AGC and a video output amplifier. The original UHF converter was adequate for the initial phases of the project, but is currently being studied with the object of improving signal quality. The crystal-controlled TV transmitter was entirely designed and built by the author. It operates on UHF Channels 23 and 24 (under experimental license KA2XBT) with a power output of about 0.25 watts using standard AM modulation. It is fully solid-state and operates from 24 volt storage batteries on board the CART vehicle. The camera is a COHU model 2000 which also operates from 24 VDC.

On the control side, the D/A converters are standard DEC units, interfaced by the author. They provide 6 d.c. levels to a modified Micro-Avionics 6-channel proportional control receiver. The transmitter encodes the levels into a pulse-width modulated pulse train. The receiver, also from Micro-Avionics, decommutates the pulse train into 6 individual pulses, of which 4 are used for control, the functions being camera azimuth, steering wheel direction, drive motor field voltage, and drive motor armature polarity (motor direction). The basic control scheme is shown in Fig. A.2.

The scheme shown is used for camera and steering control. For field voltage (drive motor speed), the potentiometer is replaced by a fixed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

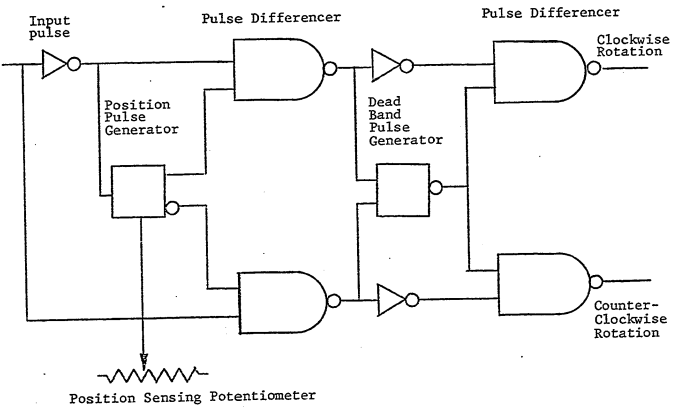


Fig. A.2 - Vehicle Control Logic

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

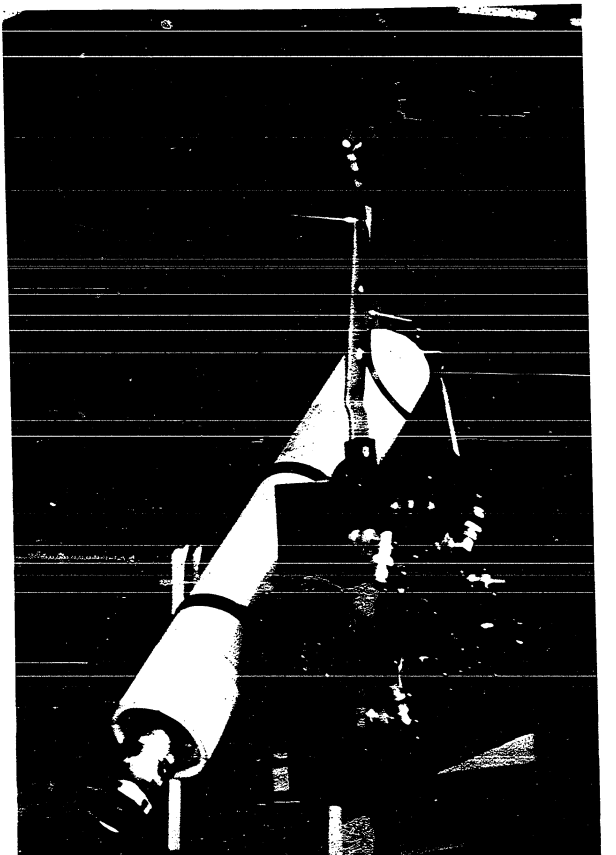


Plate 3 - Television Telemetry System

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

resistor, and one of the outputs is averaged and used to set the field voltage. For motor direction, one of the outputs is used to activate a reversing relay. The whole system, except for the radio sections and the actual relay drivers, is fabricated from integrated circuits.

Since the D/A converters have internal buffers the vehicle does not require constant control by the computer. The buffer contents will continue to control the vehicle even if the control program fails. In order to provide some protection against computer failure, the motor speed channel is equipped with a detector which clamps the speed line to a level guaranteed to stop the vehicle if the computer does not update the D/A contents frequently (about every half second). With this feature the operator is free to interrupt the program for debugging at any point without the vehicle's being damaged.

The CART itself is a battery-powered vehicle with a wheelbase (and width) of about 3 feet. It is driven by 4 electric motors, one on each wheel, with no transmission or differential. In its original form, all four wheels were steerable, so that the direction of motion of the vehicle had no relation to the vehicle body orientation. This combined with the imprecision in steering caused by such large control motions (720 degrees) made the vehicle largely unmanageable. Accordingly, the rear wheels were fixed, the control channel thus released was used to reverse the drive motor direction (formerly the vehicle could only go one way), and the front wheel travel was reduced. Another source of difficulty was the rather large variation in speed caused by the drag of the front wheels when they were turned. This occurred because in the original design the wheels were chain-driven and were always parallel when turned.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This system has been replaced by a steering arm and tie rod system similar to that used in automobiles, which comes within a few degrees of perfectly matching the Ackerman steering angles required for drag-free steering. The problem of speed variation while climbing or descending hills is still a problem though.

In conclusion let me comment that this vehicle was intended as a low-budget way of getting some experience with driving problems, and that if this research is continued; a new vehicle should be obtained. The limitations on operating time caused by battery discharge and the lack of equipment space, nonexistent suspension, and low speed of the present vehicle will make future work increasingly difficult and unrealistic.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

APPENDIX II

DISCUSSION OF SELECTED SOFTWARE DETAILS

Since the CART programs written by the author are in PDP-10 machine language, and are over 30,000 words in length, it would be senseless to try to discuss them in detail. In this appendix, three of the more interesting algorithms are discussed, one from the picture description program outlined in Chapter III, and two from the guidance control program explained in Chapter II.

1. Region Expansion Algorithms

One of the major advantages of region oriented analysis over edge oriented analysis is that regions are preserved as unified data structures at all points in the analysis, while in edge analysis, they must be constructed from assemblages of edges after all the edges are found. This is not a major problem for simple pictures, but in cluttered or poor quality pictures the problem is significant.

The major objection to region analysis has been that since pictures have much homogeneous area in them, region analyzers spend much time on data points yielding no information about the picture content (interior points of regions) and thus take longer than equivalent edge analyzers. The approach presented here shows that this problem is not an essential part of region analysis, and in fact shows that it is only necessary to reference region interior points once, the same number as required by an edge analyzer. The approach presented here is table-driven, making it even faster. A flow chart of the implementation is given in Fig. A.3, with associated tables.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

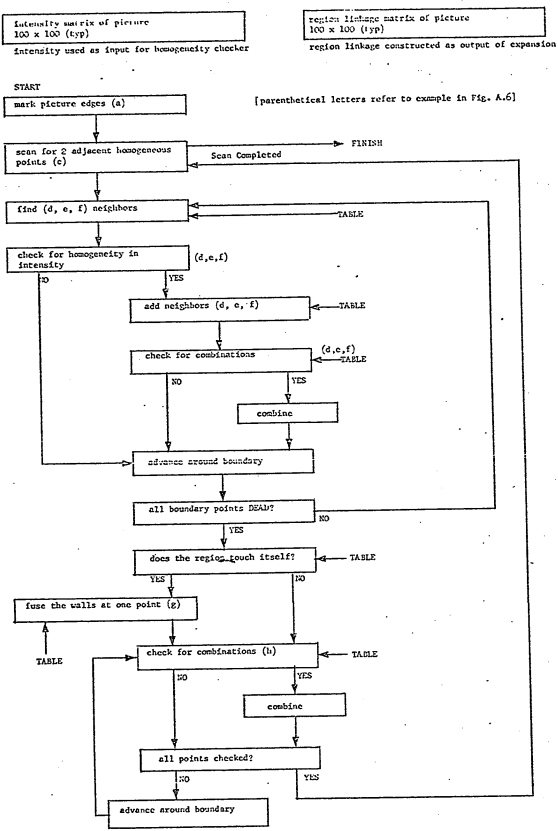


Fig. A.3 - Flow Chart of Region Expansion

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In the discussion of the implementation which follows, the mnemonics DAT1 and DAT2 refer to 6-bit field which describes the linking of each cell in the picture to its neighbors. The mnemonics VEC1 and VEC2 refer to cells containing the location in memory that the quantities DAT1 and DAT2 refer to. Normally, the cell referred to by VEC1 is linked to the cell referred to by VEC2 by the characteristics in DAT1.

Each of the 6 bits in DAT1 and DAT2 is given a mnemonic which will be used in the discussion. The mnemonics are:

BDR - the cell is on the picture boundary. This bit is used to prevent the sides of the picture from being "wrapped around" into each other.

DEAD - the cell has no neighbors suitable for incorporation into the region the cell belongs to. This bit is used to prevent rechecking the cell's neighbors when it is already known that they are losers.

LEFT - the cell is linked to its neighbor on the left. This means that if we proceed around the outline in the normal clockwise fashion with the object on our right, the next element of the object border is directly to this cell's left in the picture coordinates.

RIGHT - the cell is linked to its neighbor on the right.

UP - the cell is linked to its neighbor above.

DOWN - the cell is linked to its neighbor below.

Diagonal linkages are also permitted, and are indicated by combinations of UP/DOWN and LEFT/RIGHT bits.

Given the convention that regions are encircled in a clockwise direction, region points can only be added to the observer's left as he walks around the region. Fig. A.4(a) gives the permissable expansions.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Candidate pointed denoted ⊕ , boundary point denoted ●

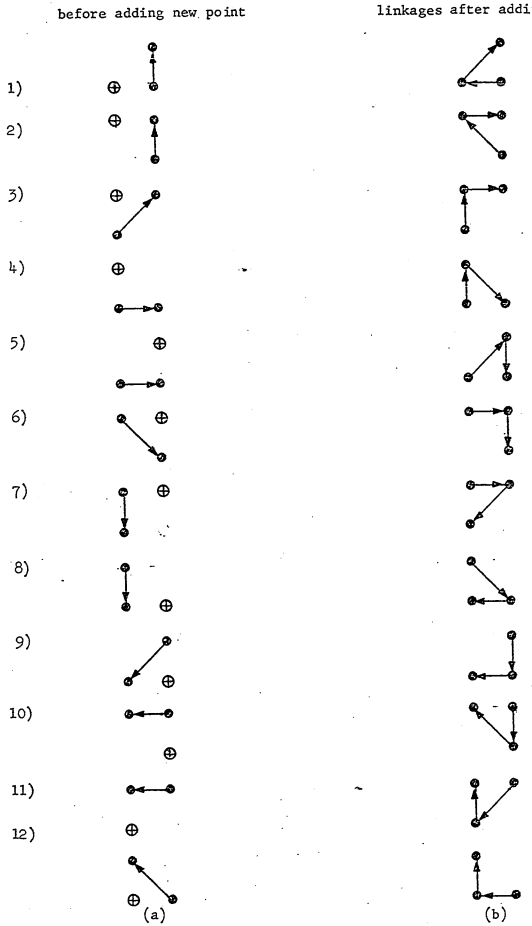


Fig. A.4 - Table of possible ways to expand a region

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The program gets these from a table and checks the relevant cells. If none of them are suitable for expanding the region (either because of inhomogeneity or because they are already claimed by another region), the DEAD bit in DAT1 is set and this cell's neighbors are not checked again. If the point was suitable, it is added to the region using another table shown in Fig. A.4(b). Now the outline is checked for possible simplifications shown in Fig. A.5. This maintains the simple geometric contour of the picture boundary and also moves points from the picture boundary into the interior where they are never referenced again.

At the conclusion of this process, the program takes one step around the picture boundary by shifting DAT2 into DAT1 and VEC2 into VEC1, and fetching the data about the cell now pointed to by DAT1 into VEC2 and DAT2. Then the whole process begins again, finally terminating when all the boundary cells of the region in question have their DEAD bits set. At this point the region is as large as it will ever get.

Since the region grows like an amoeba, when it engulfs small regions, it touches itself on the far side. This is shown in Fig. A.6(d). In order to fuse the boundaries, the program checks to see if the region is touching itself at any point. If it does, another table is used to fuse the boundaries, followed by a reapplication of the table of Fig. A.5 to remove the traces of the fusion, yielding the result of Fig. A.6(f).

Each time a point is added during this whole process the area count and other parameters of the region are updated. At the stage of Fig. A.6(f), the region is ready for final parameterization with many of the parameters partly calculated. The beauty of the whole scheme is that each of the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

boundary points denoted by , interior points by $\textcircled{i}$

potential combination after combination

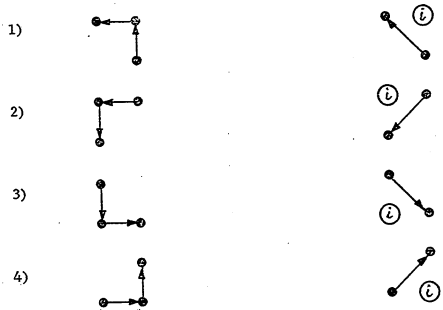
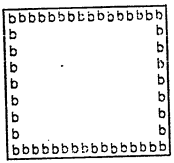
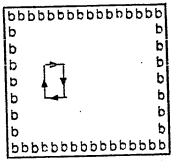


Fig. A.5 - Table of possible simplifying boundary combinations

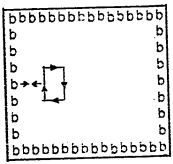
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.



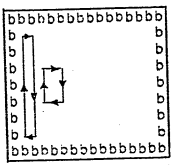
(a) mark border



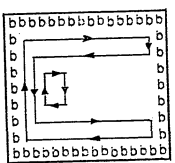
(b) expand 1st region



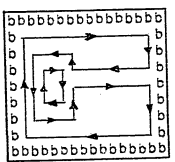
(c) create region nuciel



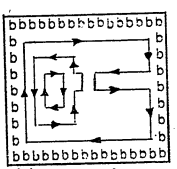
(d) expand against boundaries



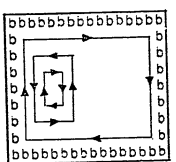
(e) region engulfs interior region



(f) region contacts itself



(g) fuse region boundaries



(h) simplify boundary

Fig. A.6-Stages in the growth of a region

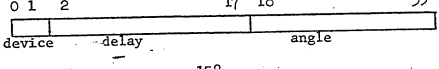
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

functions described requires a table with at most 32 entries, and no more than 10-15 machine instructions to accomplish. This makes the program extremely fast, while at the same time transparent to the programmer, an unusual feature in machine language programs.

2. Two Processor Queuing Algorithm

The A.I. Project system is split, largely for historical reasons, between a PDP-6 and a PDP-10 computer with common memory but separate I/O busses. The real-time devices are connected to the PDP-6 I/O bus, while normal computing and TV I/O processor control are done on the PDP-10. This necessitates a CART control program designed for the dual processor environment, with part of the program running on the PDP-10 CPU controlling TV reading and analysis, and another part running on the PDP-6 to output control signals to the vehicle.

Because of the time-shared nature of the PDP-10 system, there is no way to predict exactly how long the execution of the PDP-10 executed portion of the control program will take. Yet the execution of multi-part manuevers (such as turning corners and straightening out) requires precise timing of the various parts. In the CART system, this was resolved by putting the entire sequence of operations into a queue along with information setting the delay between operations. Since the operations which were to go in the queue were for the most part integer angles for various actuators with restricted travels, the full 36-bit PDP-6 word was not required to specify the angle. The operation delay and device select code could also be put in the same word. The format chosen was:



Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Since it may take longer to generate the commands than to execute them, the commands cannot be passed from the PDP-10 to the PDP-6 individually. The entire sequence of commands corresponding to some operation must be assembled and passed as a unit in order to guarantee the time relationship between the various commands. It is also desirable for the PDP-10 to be able to generate new commands while the PDP-6 is busy with the old set. This requires that the PDP-6 pick up the commands from a different area than the one the PDP-10 is using to generate them. A further requirement is that it must be possible at any time to prevent the execution of the remaining items in a queue if they have been found inappropriate after being passed to the PDP-6.

In the CART queuing scheme, the PDP-10 has a ring of four queue assembly areas into which commands are put. When a particular queue is completed, it is flagged as available to the PDP-6 and the PDP-10 makes no further reference to that area. When the PDP-6 sees the flag, it picks up the entire queue and moves it to its own area, where the commands are stored pending execution. If another queue becomes available to the PDP-6 before completion of the current queue, the current queue is immediately replaced by the new one, thus preventing any further execution of possibly outdated or erroneous commands.

The first word in a command queue is a control word whose left half is the negative of the number of words in the queue, and whose right half is a pointer to the word before the first command, i.e., the control word. This word is used by the PDP-6 hardware to extract commands from the queue. Each time a word is extracted, both halves of the control word are incremented by one. When the left goes positive, the queue is exhausted.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

This method of control makes it very easy to overlay the current queue with a new one. As part of the overlay, the current control word is overlaid also, and the extraction algorithm does not need any special code to start the new queue. Fig. A.7 is a flow chart of the PDP-6 code, which should make the process clearer. Note that only one command is extracted from the queue in each pass through the program. These passes take place at 1/6 second intervals.

Another problem introduced by the dual-processor configuration is that the timesharing system on each is independent of the continued functioning of the other. If the PDP-10 crashes, the PDP-6 will keep driving the vehicle out of the last queue received, and the operator, whose teletype is connected to the PDP-10, has no control over the vehicle.

The box at the top of Fig. A.7 labelled "CHECK STATUS" is an attempt to minimize the effect of this sort of catastrophe. If the PDP-6 program were to stop running, the hardware failsafe detector mentioned in Appendix I would stop the vehicle during the failure. However, if the PDP-6 is restarted, the CART will resume where it left off. If the operator has gone out for coffee in the meantime, the results can be disastrous. To give the operator a means of "leaving a message" for the PDP-6 not to resume running the CART, the control program on the PDP-6 gets from the time-sharing system a word describing the status of its companion module on the PDP-10. If that module is not currently runable (i.e. if it stopped because of an error, or was interrupted by the operator), the PDP-6 module immediately terminates, stopping the vehicle. Similarly, if the portion of the PDP-10 timesharing system is no longer running, it informs the CART program, which immediately terminates.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

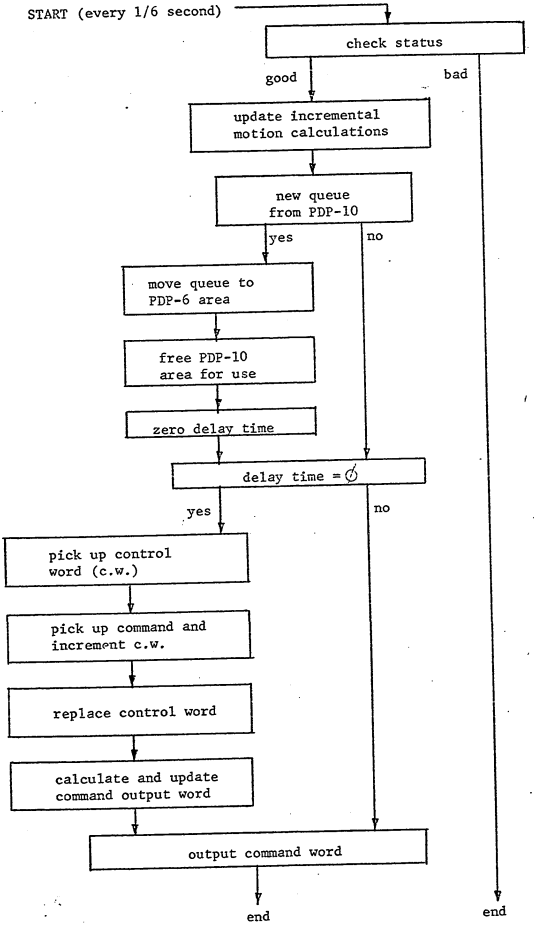


Fig. A.7 - PDP-6 Control Module

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In this way, the operator can ensure that the vehicle will stop by merely interrupting execution of the PDP-10 program. This does not require proper functioning of the queuing algorithm or of any part of the PDP-10 code - interruption of execution is obtained directly from the time-sharing Monitor. Thus the operator can introduce new algorithms into the program without worrying about "bugs" causing damage to the hardware. In an experimental system such as this one, this freedom is essential to progress, and the vehicle has been saved innumerable times from serious accidents by this feature.

3. Fast Convolution Algorithm

Computer convolution is ordinarily too time-consuming a method to use in real-time control applications. For a field of width k and a convolution mask of width m, each iteration of the mask involves m fetches, m multiplications, m-1 additions, and 1 store. For the total field, this becomes m(k-m+1) fetches, m(k-m+1) multiplications, m-1(k-m+1) additions, and k-m+1 stores. On the PDP-10, fetches take 2.43 microseconds, fixed point multiply takes 9.81 us, fixed add takes 2.75 us, and store takes 2.58 us. If k=256 and m=50, then the total to calculate the convolution function is 155.1 millisconds. This is for only one line of visual data.

However, as we have seen in Chapter II, the mask for a line or edge is uniform over its nonzero area, and thus can be calculated much faster. Not only are no multiplications required, but the value of the convolution function can be changed as we move along the line by one addition of the value immediately to the right and a subtraction of the value at the left end. For a box correlation in a field of k elements with a box of width m,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

1 fetch, k-1 additions, k-m subtractions and k-m+1 stores are required. Using the same values as the previous calculation, the time required is 1.8 milliseconds, a saving of over 80:1 over the previous figure. The correlation mask for an edge is a composite of two box functions with opposite sign, but this can be produced by a subtraction during the search for the maximum of the correlation function, and so does not require an additional pass over the data.

Once the correlation function is determined, it must be searched for a maximum. The search is made more complicated by the fact that the function may have several maxima from various bright areas in the picture, only one of which is part of a line. Chapter II describes the way in which the maxima are checked. What will be presented here is the way that repetitive maxima searches are carried out. Suppose that the correlation function has been searched and the largest maximum has been found, but it has been rejected by the line locator. We must somehow mask out the maximum so that we don't find it again in subsequent searches. At the same time we must preserve the data values of the correlation function so we can still use them in comparisons. This prevents us from zeroing out the rejected region. If we negate the rejected values, they will surely never be selected as maximums, but since all the values in the original correlation function were positive, we can recover the values for comparison by fetching the absolute values. Since this is a PDP-10 machine instruction, no extra overhead is involved. In order to avoid making one pass over the convolution function per point rejected, an area around the point of width m is rejected at the same time, so only k/m passes need be made if the correlation function contains no

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

appropriate maxima.

The alternate method for scanning the convolution function would be to pick out the k/m most likely maxima all in one pass, and check them in sequence. There is vastly more bookwork involved in this, and in the average case only one or two passes are required anyway. It is for this reason that multiple passes are used.

The computing time for the whole process is sufficiently small that the prime source of delay is the time required to input the TV picture, especially when the data channel is busy with disk transfers and the TV has to wait. Even when the disk is not active, waiting for the appropriate scan lines on the TV can take as long as 1/30 second (33 milliseconds).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

BIBLIOGRAPHY

1. Zahn, C.T., "Two-dimensional Pattern Description and Recognition via Curvaturepoints", Stanford Linear Accelerator Report No. 70, December 1966.

2. Shaw, Alan C., "A Proposed Language for the Formal Description of Pictures", SLAC Graphics Study Group, Memo No. 28, February 1967.

3. Wichman, William M., "Use of Optical Feedback in the Computer Control of an Arm", Stanford Artificial Intelligence Project Memo No. 56, August 1967.

4. Nilsson, Nils J., "A Mobile Robot: An Application of Artificial Intelligence Techniques", Stanford Research Institute, January 1969.

5. Sutro, L., Kilmer, W., "Assembly of Computers to Command and Control a Robot", AFIPS Conference Proceedings, Spring 1969.

6. Fenton, R.E., Cosgriff, R.L., Olson, K.W., Blackwell, C.M., "One Approach to Highway Automation", Proceedings of the IEEE, Vol. 56, No. 4, April 1968.

7. Tenenbaum, J.M., "Accommodation in Computer Vision", Stanford Artificial Intelligence Project Memo No. AIM-134, October 1970.

8. Kelly, M.D., "Visual Identification of People by Computer", Stanford Artificial Intelligence Project Memo No. AIM-130, August 1970.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.