Model-Directed Mobile Robot Navigation

C. Fennema, A. Hanson, E. Riseman
J.R. Beveridge and R. Kumar

COINS TR 90-42

June 1990

Model-Directed Mobile Robot Navigation[1]

Claude Fennema, Allen Hanson, Edward Riseman
J. Ross Beveridge and Rakesh Kumar

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01002

[1]This work was supported in part by the Defense Advanced Research Projects Agency under contract numbers F30602-87-C-0140, DACA76-85-C-0008, and DACA76-86-C-0015, and by the National Science Foundation under grant number DCR-8500332. The material in this paper was partially presented at the DARPA Image Understanding Workshop in Palo Alto, CA, May 1989.

Contents

Abstract

The UMass Mobile Robot Project is investigating the problem of intelligent navigation of an autonomous robot vehicle. Model-based processing of the visual sensory data is the primary mechanism used for controlling movement through the environment, measuring progress towards a given goal, and avoiding obstacles. Goal-oriented navigation takes place through a partially modeled, unchanging environment which contains no unmodelled obstacles; this simplified environment provides a foundation for research in more complicated domains.

The navigation system integrates perception, planning, and execution of actions. Of particular importance is that the planning processes are reactive and reason about landmarks that should be perceived at various stages of task execution. Correspondence between image features and expected landmark locations are used at several abstraction levels to ensure proper plan execution. This paper describes this system and some experiments which demonstrate the performance of its components.

1 Introduction

The UMass Mobile Robot project is investigating the problem of enabling a mobile automaton to navigate intelligently through indoor and outdoor environments. Model-based processing of the visual sensory data is the primary mechanism used for controlling movement through the environment, measuring progress towards a goal, and avoiding obstacles. At the foundation of our work is the premise that higher-level vision beyond the first stages of sensory processing will greatly benefit from, and in many cases require, the use of knowledge and models of objects in the environment.

Our mobile robot, called Harvey,[2] is a Denning Mobile Robotics platform ultimately intended to navigate through offices, hallways, and university grounds as it carries out commands such as "Fetch the book" or "Bring this to Allen". This is a rather formidable task; consequently a research plan has been developed that will be carried out in stages of increasing generality and functionality. In the early phases of this research, we wish to balance generality against achievable research goals. Consequently, the initial experiments focus on robust goal-oriented visually guided navigation through a partially-modeled, unchanging environment that does not contain any unmodelled obstacles.

If robust autonomous navigation can be achieved in this restricted domain, then a variety of challenging problems can be considered as the constraints on the assumed knowledge about the environment are relaxed. These problems include: navigation in a partially known environment with stationary unmodeled objects, navigation in the presence of independently moving objects, and exploratory navigation in unknown environments to learn the models necessary to support future model-directed navigation. The focus of this paper, however, is on robust navigation in a partially-modelled environment and the results of initial experiments in this problem domain.

[2]Named after Harvey Wallbanger cocktail, which also characterizes some of its early behaviour.

1.1 Related Mobile Robot Research

We begin with a brief survey of previous mobile robot research; other relevant research will be addressed in the sections discussing particular system modules. The Carnegie-Mellon NAVLAB [22,33] and the Martin-Marietta ALV [30] are two robotic vehicle systems designated to move down a path or road or navigate through off-road terrain. In both systems, the processing has been restricted to short term goals, such as controlling the vehicle relative to the sides of the road, or avoidance of major obstacles such as trees. Recent demonstrations of these systems have been quite interesting; but, unlike the project discussed here, these sytems used a laser range sensor providing depth information which played a significant role in the obstacle avoidance capabilities.

Dickmans and Graefe [9,10] have developed techniques for using image features in a real time feedback control loop to control the motion of a car on the Autobahn. In the system described here, their techniques could serve as part of the function we term "action level servoing". Our approach accomplishes servoing in a similar way by tracking image features, but here the tracking features are constructed from landmarks which have been selected from a knowledge base.

Brooks [5] has shown an unusual integration of low-level behaviors and motor activities which allow a relatively inexpensive robot to wander in an unknown environment. This work, however, has not yet focused on the achievement of higher-level goal-oriented navigation tasks, and does not make use of models of the environment.

Recently, Faugeras [34] used more sophisticated vision algorithms involving stereo to derive depth in an office scene. Depth information was extracted from a stereo pair, the robot was moved some distance, and a second stereo pair was used to derive depth and the associated motion. Again, this effort does not represent a full robot navigation system and made no use of high level models.

Herman [17,18] has described an extensive system, called MAUV, designed to control multiple undersea vehicles in the execution of intelligent, autonomous, cooperative behavior. Behavior in this system is the result of schema based planning in a task subdivision hierarchy. Unlike our system, MAUV produces action by executing steps as outlined in complete plan graphs, rather than

developing plans incrementally. In addition, MAUV separates the sensory and planning activities into separate processes, using the model as a buffer between them. The system described in the following integrates planning, action and perception.

Due to the complexity of visual perception, autonomous navigation projects such as those cited have utilized only limited visual processing, either in terms of the features extracted from the environment, or the modeled set of objects to be recognized in the environment, or both. This is not meant to be a criticism, but rather serves as an observation for the reader who does not recognize the extreme complexity of the problems of vision and autonomous navigation in complex outdoor domains.

1.2 Overview of System Modules

The processing modules that provide the basic functional capabilities for the mobile robot system are briefly outlined below. There are many possible control strategies and system organizations that can be imposed on top of these modules to support effective mobile robot navigation. In Section 2, we briefly outline one such control strategy.

Modelling the 3D Environment [7,8] - Geometer is a solid modelling package that was jointly developed at the University of Massachusetts and General Electric Corp. Designed with vision in mind, this general CAD system provides tools for representing knowledge of shape in an annotatable hierarchy.

Plan Sketching [13,15]- Navigational goals are decomposed by a plan sketcher into a sequence of less abstract subgoals and milestones. This sequence, called a plan sketch because it is only partially developed, is used to reason about what action to take next. Such sketches are tentative and change frequently.

Reasoning about Action [13,15] - Harvey does not generate detailed plans. To decide what action to take next, plan sketches are developed depth-first, with less detail away from the current

location; task failures trigger changes in plan sketches at various levels. Goals result in action by means of a repetition of two operations: recognize milestone and execute subgoal. If the subgoal to be executed is a primitive subgoal then action results, otherwise a more detailed plan sketch for that goal is proposed. Each milestone is constructed from a set of perceivable 3D landmarks derived from the model. Finding the expected projection of the landmark in the image signifies a successful completion of the associated action.

Executing Primitive Subgoals [13,15]- When it has been determined that a subgoal can be carried out with no further reasoning it is executed in a closed loop manner using what we have called "action level servoing". Landmarks selected from the model are used to steer the robot. Corrections to the motor system are made when the image of a landmark moves differently from what is expected.

2D Line Model Matcher [3,4] - This module finds a best match and fit of a given 2D line model to a subset of data line segments that may have been fragmented, skewed, omitted, etc. during low-level processing. A search through the plausible symbolic correspondences between model and data lines is performed, and the optimum 2D translational and rotational fit for each is computed as a closed-form solution.

3D Pose Refinement [23,24] - Given correspondences between a set of points and lines in a 3D model and a 2D image, the 3D camera location and orientation is computed as an optimization procedure. In addition, uncertainty in the output parameters as a function of the variance of the noise in the input parameters is provided.

In addition to these modules, several basic vision modules have been developed. These modules include a fast line finder [21] derived from a straight line algorithm developed by Burns [6], a histogram based region segmentation algorithm [2], and a template correlation mechanism [15].

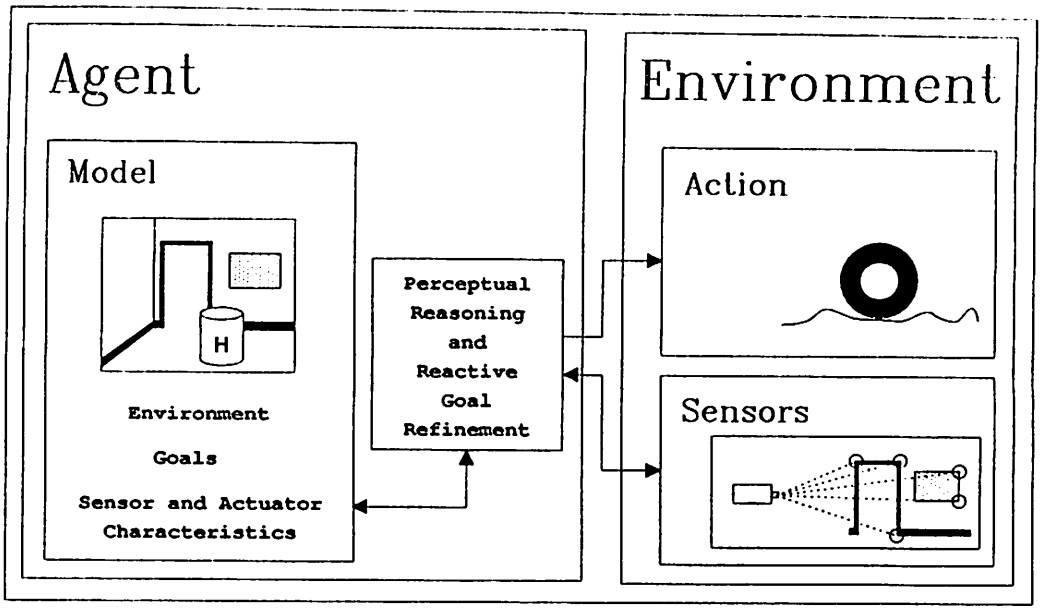


Figure 1: Harvey's navigational system uses sensors as a blind man uses a cane. As the system reasons about its actions, it also reasons about how to selectively probe the environment in order to verify that its model and reality are in agreement.

2 System Overview

The robot's system architecture is designed in such a way that planning, perception and action are integrated in a fine grained, interwoven control structure. The system reasons incrementally about the actions necessary to accomplish its long and short range goals and about what perceptions should result from the execution of those actions. Actions are based entirely upon the system's internal model. When an action decision is made, sensor data is compared with expected perceptions as the action takes place. Differences from expectations may modify the execution of the action, and possibly short range or long range goals.

Perception is used as a blind man uses his cane. Hypotheses about the environment are made and the cane is used to probe to see if these hypotheses are consistent with reality. Sensing is parsimonious and planned. The model and the reasoning which arises from it are considered valid as long as sensory probes are satisfied. Figure 1 illustrates this concept. Using knowledge of the sensor and actuator characteristics the reasoning system selects key features from the environmental model to serve as probes. In Figure 1 the visual sensor is being used to probe for some corners around the door and the blackboard.

This *verification-oriented vision* is used to modify robot behavior in a three level control strategy we call "perceptual servoing" (see Figure 2). At the lowest level, each *"primitive" action* is controlled using *"action level" perceptual servoing*: the vehicle is steered using information about how a selected landmark should move in the sensed image as action takes place. At a higher level,

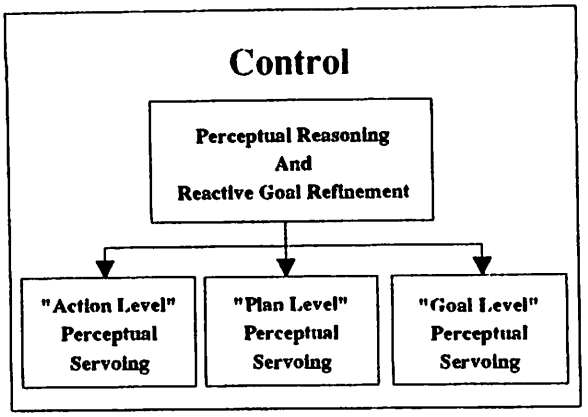


Figure 2: Harvey's navigation system reasons about routes, actions and perception. Expectations of what is to be perceived are generated to monitor each action, to measure progress toward the goal and to determine the robot's location should it get lost.

"plan -level" perceptual servoing monitors the vehicle's progress . As the system reasons about its actions, it creates plan sketches which include milestones. These milestones describe perceptual expectations which should be met when the robot reaches a certain point in its journey. Minor deviations from these expectations cause plan adjustments. Finally, should deviations from milestone expectations be large or should the robot get "lost", *"goal-level" perceptual servoing* generates a set of perceptual probes to determine the vehicle's current location. Once this position is determined, a new top level goal is established and a new set of plan sketches are generated.

3 The System Model

In this section we describe the model used by the system reasoning processes. The reasoning and control processes themselves are outlined in Sections 4 and 5. We begin by discussing some of the factors which motivate the design of our model.

3.1 Motivation

The long range objective of this project is to construct a mobile automaton which can perform the "go fetch" task in real time and in real environments. Because of the computational complexities inherent in perception, planning, and reasoning processes, the system must structure its knowledge for efficiency. Processes must be highly focussed on relevant information and the system as a whole must be highly goal-directed. Finally, the system must be capable of resolving the inevitable discrepancies between its internal model and the sensed environment in such a way that its overall goal is not compromised.

When reasoning about space it is important to be able to access relevant information quickly and to ignore information which is too detailed, too global, too distant or just irrelevant. When reasoning about a path through a laboratory, for example, the system should be able to ignore facts

about the objects in the top right hand drawer of the desk, facts about the relationship between the earth and saturn, and facts about the streets of San Francisco. The representation of space presented in Section 3.4 offers focusing mechanisms by providing various abstraction hierarchies and encapsulating mechanisms.

When interacting with space (action and perception), it is also important to be able to cope with imprecision. A one percent error in the estimate of the position of the robot relative to a coordinate system centered somewhere in Chicago could place Harvey in Northampton (a nearby town), rather than in Amherst. In Harvey's model, locations are described in terms of a hierarchically organized system of neighborhoods, each with a local coordinate system. The neighborhoods, which we call *locales*, are organized into a contained-by hierarchy which offers a topological method of specifying location: a point can be described as known to be within a particular member of the neighborhood system. Its location within that neighborhood can be further described in terms of the neighborhood's local coordinate system.

The philosophy of localization within this system is to use perception to identify a locale that is low in the hierarchy which contains the location and then to describe the location in terms of the locale's coordinate system. This approach helps to cope with the inevitable global inaccuracies. If Harvey were to travel from Amherst to Northampton, it would pass from one neighborhood into another. On a large scale this is a statement of a change in location which relates the robot's actions to high level goals in a way which is more natural than using a global coordinate system. This hierarchical sytem of locales is described in Section 3.4.

3.2 Modeling Perception and Action

For this system to function it is necessary that it have a model of certain facts about its sensors and actuators. The physical realization of an action rarely conforms to what intended. Surface irregularities, the elasticity of the vehicle wheels, and differences in wheel sizes all cause the vehicle to deviate from an intended course. Some knowledge of the characterization of these deviations is essential to effective control of the vehicle. This knowledge simplifies the task of interpreting the changes in the sensed imagery as the vechicle moves and provides necessary information for determining corrective actions. A model of sensor characteristics is also important. Modeling the sensor projection parameters provides a means for relating image data to physical dimensions and a characterization of photometric parameters is essential for constructing perceputal probes.

To simplify the actuator model we have restricted the suite of primitive motor actions to include only two types of intended motion: straight line forward motion and rotation about the robot's axis.

All motion is ultimately decomposed into a series of such motions. As mentioned above, however, the flexibility of the vehicle tires and the unevenness of the traveling surface, cause forward motion to drift significantly (see Table 1 in Section 5). Fortunately, for forward motions of under five feet, this drift can be modeled as a circular arc, tangent to the intended line of motion at the starting point. A simple model such as this is essential for the type of control described in Section 5. Were the actual motion to be curved in an "S" shape, for example, situations could arise where positional and heading errors could cancel their effects on how tracking points are imaged. This would make the "action-level" perceptual servoing less reliable. Intended rotations do result in some translational motion, but fortunately this is sufficiently small to ignore.

The sensor model is more complex and is one of our ongoing research topics. Currently the model consists of: a set of camera parameters including focal length, aspect ratio, and image size, a set mathematical transforms including projection and clipping; and the set of low-level vision modules described in Section 1.2.

3.3 Modeling the Task

The task model represents what is to be done and how progress toward this goal is to be measured. The basic elements of this representation are goals and milestones. Goals are represented using conceptual dependency notation [32]. This choice reflects the possibility of directing the navigational system via a command interpreter based on natural language. The goal to move from one location to another is represented as:

(ptrans laboratory allen's-office)

In natural language such a goal would correspond to the goal:

"Go from the laboratory to allen's office"

Milestones are descriptions of perceptual events which should be seen when a goal has been satisfied. They form the basis for measuring progress. The milestone corresponding to the above goal might be:

"A large black and brown desk should be to the right".

Goals and milestones are collected into plan sketches. Plan sketches represent tentative plans with embedded milestones. They are represented as a list:

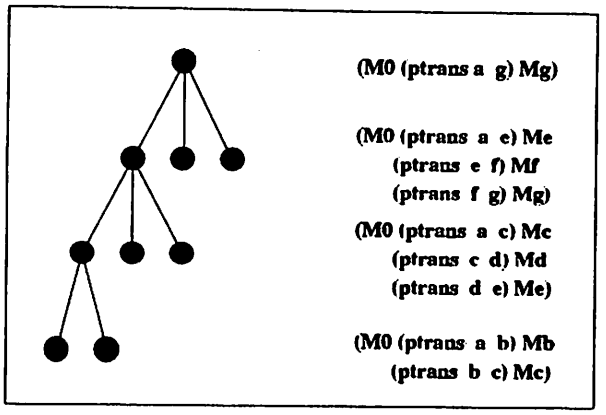


Figure 3: A plan sketch hierarchy. Plan sketches are refined in depth-first fashion until a primitive goal is encountered. The root node of this hierarchy is the original goal. The first level is a complete, but very abstract plan. The unexpanded nodes in this diagram are goals yet to be satisfied.

(M0 (ptrans a b) M1 (ptrans b c) ... (ptrans f g) Mg)

In this list each Mk is a milestone. M0 is a precondition milestone which must be satisfied before the plan sketch can be considered meaningful. [Each Mk for k $\succ$ 0 must be true if the goal preceding it has been satisfied]. These lists are called plan sketches, rather than plans, because they are approximate, abstract and are of a tenative nature. Goals in a plan sketch often refer to large scale movement and these goals are often modified during the execution of a navigation task.

The task itself is represented as a tree (Figure 3). The root node represents the current top level goal. The children of each node in the tree are the subgoals of the current plan sketch designed to accomplish the goal represented by that node. In the control scheme described in Section 4, these trees are expanded in such a way that at any one time only one node at each level is expanded into its children. What results is a hierarchy of plan sketches. In this hierarchy the collection of nodes at a particular level represent goals in the currently active plan sketch at that level.

3.4 Modeling the Environment

The system begins with an accurate, but incomplete, model of the world implemented as a graph or network, described later in this section. Portions of the interior of our building, the UMass Graduate Research Center, as well as a portion of the surrounding campus were modelled as volumes using planar surface patches, and embedded in the locale hierarchy described below. Figure 4 is a projection of a portion of the model of the UMass Graduate Research Center hallway used in the experiments.

The construction of an accurate 3D model of an environment is a fairly difficult job. The first attempt involved digitizing data from engineering blueprints using a bit pad (digitizing tablet). This method is quite error prone given the spatial resolution of the bit pad, since the blueprints

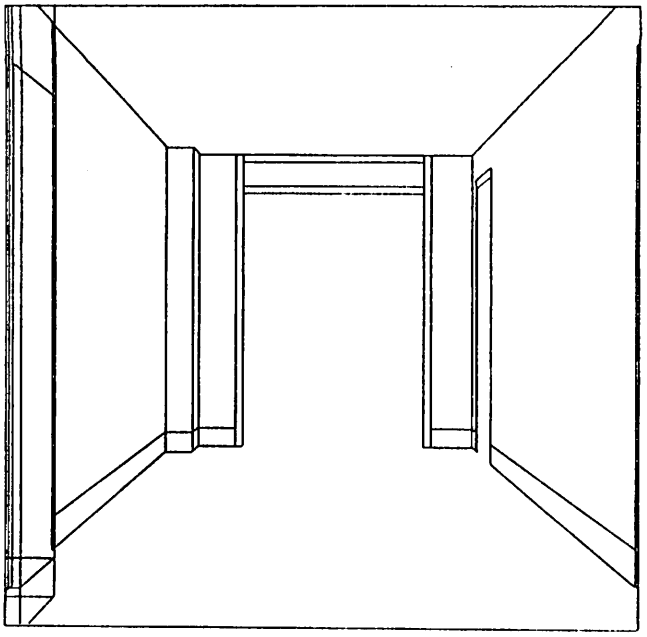


Figure 4: The model of the environment is incomplete, but positionally accurate. All objects are modeled but many details may be absent. This scene, which is projected from the model, shows walls, baseboards, moulding and pillars. Details like wall sockets and posters are not always in the model.

were drawn to a scale of 40 feet to an inch We found errors of up to 10 feet in the 3D model constructed in this manner. In the second attempt, theodolites were used to survey the landmarks. This method, while accurate, is very time consuming. As a check, some of the theodolite data was verified by direct measurement. On the average, the measured distances matched with the surveyed distances to within 0.2 feet.

The environment is represented as a graph structure which captures the key topological, geometric, and physical properties of the spatial entities involved. This network describes space in terms of a collection of "locales": spatial entities representing objects, buildings, parking lots, free space, etc. A locale is a parcel of space which has semantic significance to the navigation problem. Each locale is represented as a node in the network; arcs relate locales by means of part/subpart relations as well as by the allowable transitions between the locales (see Figure 5).

The shape and spatial extent of each locale is represented by a description of its surfaces. Currently surfaces are described in terms of planar surface patches called faces (see Figure 6). Like locales, faces are represented as network nodes which contain information about the properties of the face. Arcs describe its relationship to other faces as well as to the locale of which it is a part

The geometric properties of each face are represented in terms of lines and points, which are similiarly represented. Global physical properties of a face, such as area, are kept on its property list. Visual properties, which may vary over the surface of the face, are described in terms of regions (Figure 7). Faces may have two sets of regions, corresponding to the two sides of the face. The two sets of regions account for the possible differences in appearance between two sides of a wall,

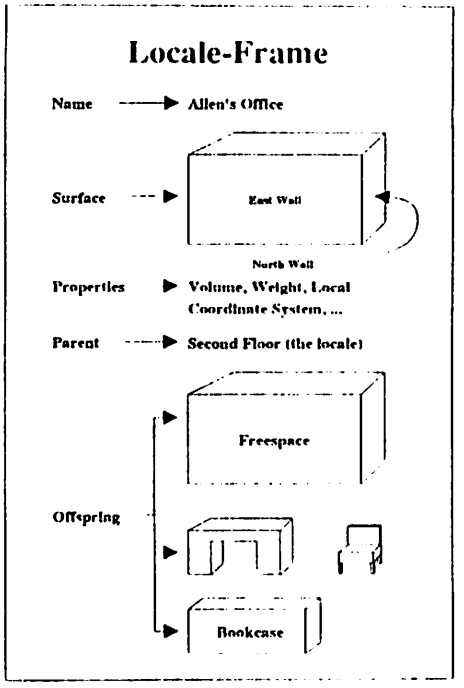


Figure 5: All 3D space entities are represented as locales. Locales are implemented as frames with slots which organize the topological, geometric, and physical properties of the spatial entities they represent.

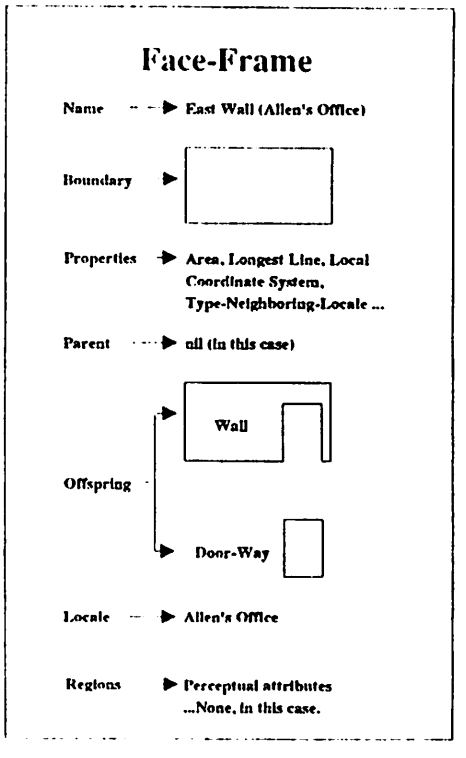


Figure 6: Locale surfaces are described in terms of planar faces. Faces, like locales, are represented as frames. The properties of a face include information about its type (door-way or barrier) and, if it is a door-way, what locale it offers passage into. The Face-Frame for the second offspring of the face represented here, for example, would be of type door-way, offering passage into the hallway locale.

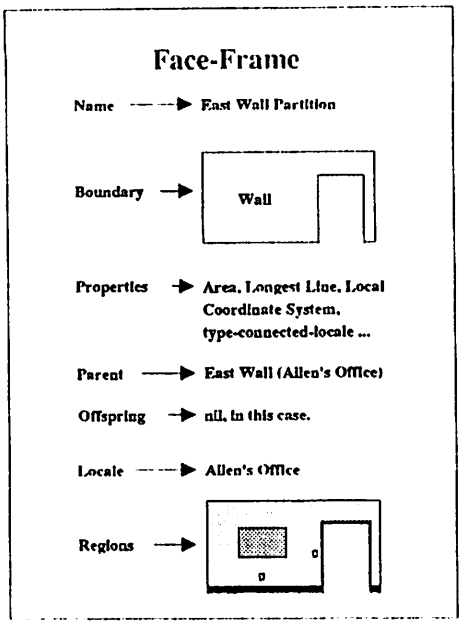


Figure 7: The appearance of each face is described in terms of a collection of regions. Each region represents an area of the face which has uniform appearance properties (color, texture, etc.).

for example.

The locale network provides quick access to the geometric, topological and physical properties of the individual locales. The network also implements a number of useful abstraction hierarchies which offer focusing mechanisms for reasoning about actions and perception. Figure 8, for example, shows the locales organized into a "contained by" hierarchy. This portion of the network shows the Research Center as containing six sublocales: four floors, an elevator (shaft) and a stair. The other arcs in this figure represent doorways between the locales they connect. Doorways, such as the offspring locale in Figure 6, indicate the locale they open into via the type-neighboring-locale property on the property list (also shown in Figure 6). When reasoning about what actions to take to satisfy a goal (described in the next section), various levels of this hierarchy are used.

4 Reasoning About Actions

Each task given to the robot is translated by a command interpreter and problem solver which ultimately produces a set of navigational goals. The execution of these goals is accomplished by a tight interweaving of planning, perception, and action. It is orchestrated by a dynamic planning and execution scheme [15,13] called *"plan-and-monitor"*, which reasons about goals and decides what action to take next. This subsystem works with the plan sketch hierarchy described in Section 3.2, which represents the current view of how the task can be accomplished. This hierarchy is developed in a depth first manner shown pictorially in Figure 9; this figure corresponds to a plan sketch hierarchy such as the one shown in Figure 3. The milestones embedded in each plan sketch are perceptual events which are used to verify that action and reasoning are in agreement;

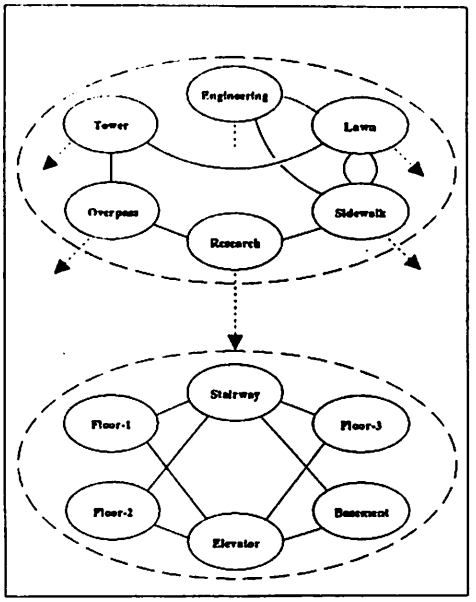


Figure 8: Locales are organized into a contained-by hierarchy. In this figure the research locale at the top level is partitioned into six sub-locales at the lower level: four floors, an elevator and a stairwell. The arcs at each level represent the passageways or door-ways between the locales they join.

consequently, they can be used to measure progress toward task completion.

Milestones typically specify 3D landmarks and their expected location with respect to the robot at the completion of the appropriate phase of a plan sketch. As motor actions take place, milestones are verified (usually visually) before the next goal of the plan can be undertaken. This ensures that the robot's actual location in the world agrees with its intended location with respect to its world model; this in turn ensures that the next goal in the plan-sketch hierarchy is applicable. For example, if the sequence of milestones up to M_i have been perceptually verified to be in the proper position in the image (i.e. within the acceptable error bounds), then we assume that goals $G_o \cdots G_i$ have been satisfied, and it is appropriate to consider goal G_{i+1}. If M_i cannot be verified, then the plan-sketch hierarchy must be modified. In this way milestones allow the progress of the plan to be monitored. Whenever a milestone cannot be verified some replanning must take place before the next subgoal can be undertaken [14].

The plan-and-monitor executive directs planning, perception, and execution in such a way as to dynamically modify and refine the plan-sketch hierarchy to fit the actual results of each action and the details of the perceived environment. The principal activities involved in this process are: creating plan sketches, milestone recognition, determination of location, and execution of primitive motor actions. This interweaving of perception, planning and action makes specific what task is expected of perception, and provides a means for focusing the knowledge available for that purpose. The result is a distribution of perception and perceptual reasoning into all aspects of navigation. Route planning uses perceptual reasoning to select appropriate perceptual milestones; plan progress

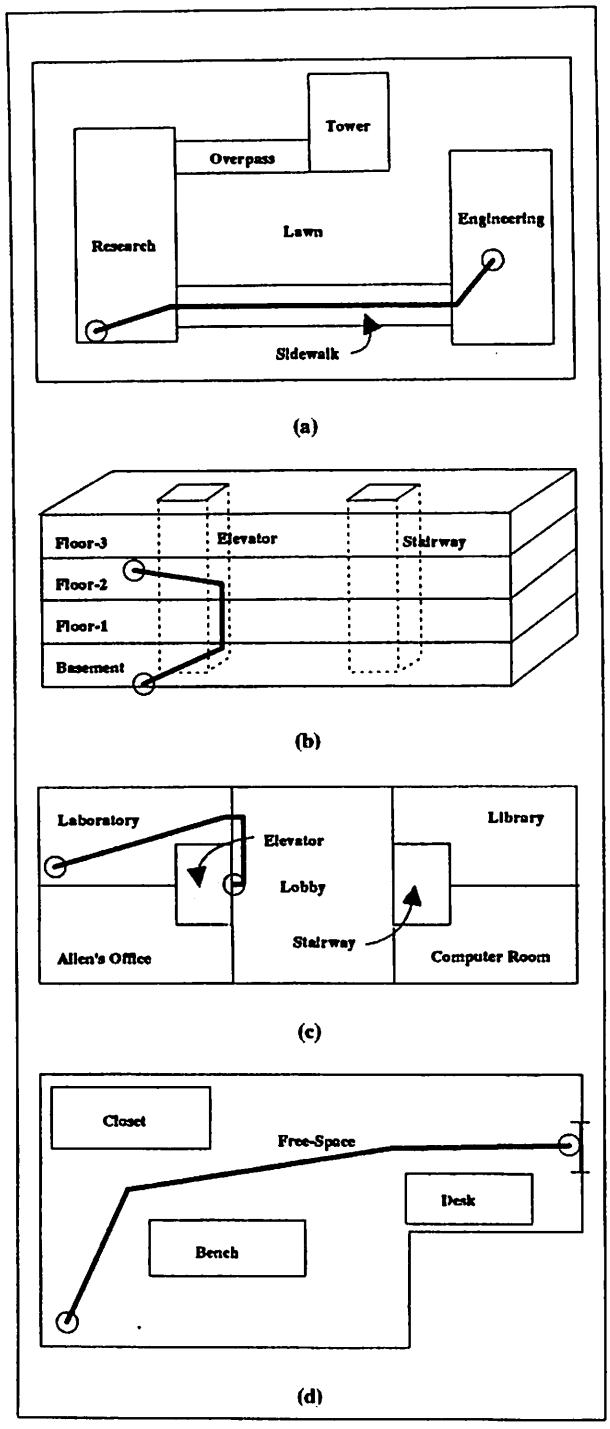


Figure 9: Given a goal to move from the laboratory in the Research Center to an office in Engineering, a plan sketch is proposed at a very abstract level. (a) Using the connectivity network shown in Figure 8, an attempt to satisfy the first goal in this plan sketch forces a more detailed plan sketch development in (b), (c), and (d). In (d) the first subgoal is a primitive one so action begins.

is measured using perception; perception is used to relocate the robot when a milestone is not recognized; and during the execution of primitive actions, low-level perceptual feedback is used to keep the robot on the expected trajectory. The different levels of control all use model-directed vision and compare what is sensed to what is expected, issuing corrective commands to minimize any difference.

The next section describes the "action-level" perceptual servoing used during execution of individual actions. Section 6 describes a method for matching models to landmarks used for milestone-recognition.

5 Executing Primitive Subgoals: Action-Level Perceptual Servoing

Navigation goals are ultimately translated into primitive subgoals which can be executed by the robot vehicle with no further subgoal development; for the current platform, the single primitive subgoal corresponds to an unobstructed straight line path between two points. A path such as this can be achieved by a (TURN angle) followed by a (MOVE distance). Even when carrying out primitive subgoals such as this, however, execution errors are probable. As our Denning robot platform performs an open-loop turn (i.e. no feedback control), the rubber tires produce a sideways motion or "skittering". As the robot moves forward along an environmental surface, a slippery spot, unevenness of the surface, or even a bulge in a tire may throw it off course, causing inaccurate execution. We reduce this error by servoing on prominent visual features in the environment. This we call "action-level" perceptual servoing. Before each action begins, landmarks are selected from the model for tracking. This is done by analyzing the model to select landmarks which should be visible to the robot from its present position. Image correlation is used to track each landmark, so it is important that they be distinctive. The action is then carried out incrementally, comparing the projected and actual locations of the landmark images (see Figure 10). New tracking landmarks are chosen as old ones move out of view. By measuring the discrepancy between where the landmarks should be and where they actually are, it is possible to determine the corrective action required to bring the positions into agreement. This traditional servo control has the effect of locking the robot onto a trajectory which improves the accuracy of the primitive actions over that which would be obtained without servoing.

In order to determine the usefulness of servoing, a simple version was implemented that used correlation to measure the deviation of actual motion from intended motion. Numerous exper-

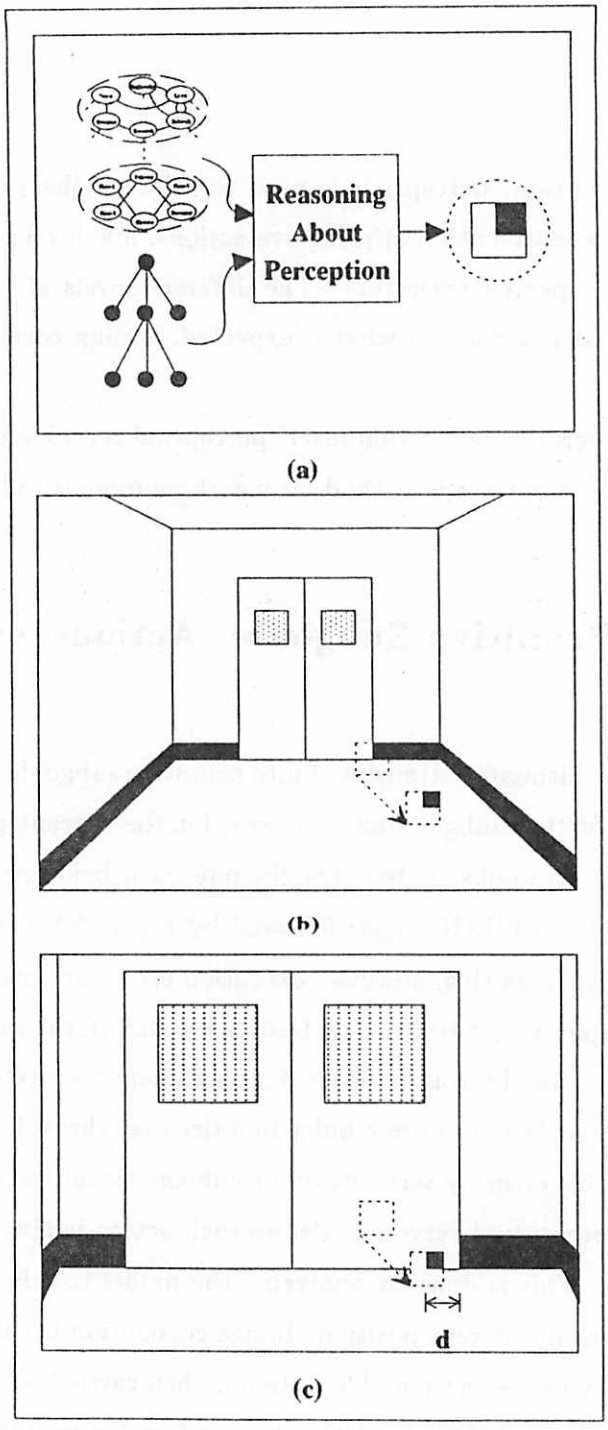


Figure 10: Before each action begins, a landmark feature is selected. (a) using knowledge of the model and the task. Action is then carried out incrementally. Before each increment is complete the motion of the landmark in the image is predicted. (b) Any difference between this projection and the location of the actual landmark image, (c) are used to determine what corrections to make.

iments, both with and without correlation servoing, have been run. In the experiments Harvey was to move along a straight line 40 feet long, marked on the floor of a hallway in our building. The robot's goal was to move down the corridor directly towards the landmark. To determine course deviation, a two foot increment was established between correlations and at the end of each increment the robot's deviation from the marked line was measured.

The results in Table 1 and Figure 11 represent "the best" single experiment in the sense that the unservoed results represent the smallest deviations encountered during the trials. The z-axis in the table refers to the intended straight line path of the vehicle, with z=0 defined as the starting location. The x-axis is a line perpendicular to the z-axis and pointing to the right. Total distance traveled in the unservoed case is z-unservoed and deviation is from the intended straight line x-unservoed. Even after a rather painstaking initialization procedure to align and orient the vehicle, it wandered over two inches from the line during a 20 foot open-loop motion. Other runs resulted in as much as a foot deviation in this unservoed mode. Most of the trials in unservoed mode were stopped at around 20 feet because the vehicle was significantly off course and the total deviation was increasing. In contrast, while in servoing mode the vehicle stayed within .3 inch of the line for 38 feet. These results are very encouraging and support the use of action-level servoing to increase the reliability of execution of primitive actions. It is worth noting that in both experiments the actual distance covered was somewhat less than the intended distance. It is consistently short by a constant factor (to 3 decimal places), due to calibration problems with the hardware.

Table 1. Results from one experiment
(All measurements are in inches)

intended-z	unservoed-z	servoed-z	intended-x	unservoed-x	servoed-x
24.	22.6	22.8	0.0	0.0	+0.13
48.	45.5	45.7	0.0	-0.3	+0.13
72.	68.3	68.5	0.0	-0.4	+0.13
96.	90.9	91.3	0.0	-0.6	+0.13
120.	114.2	114.3	0.0	-0.7	+0.06
144.	136.3	136.9	0.0	-1.1	0.00
168.	158.2	159.5	0.0	-1.3	-0.13
192.	181.8	182.2	0.0	-1.8	-0.13
216.	204.6	205.0	0.0	-2.0	-0.38
240.	228.3	227.7	0.0	-2.1	-0.25
—	—	—	—	—	—
480.	—	456.0	0.0	—	0.0

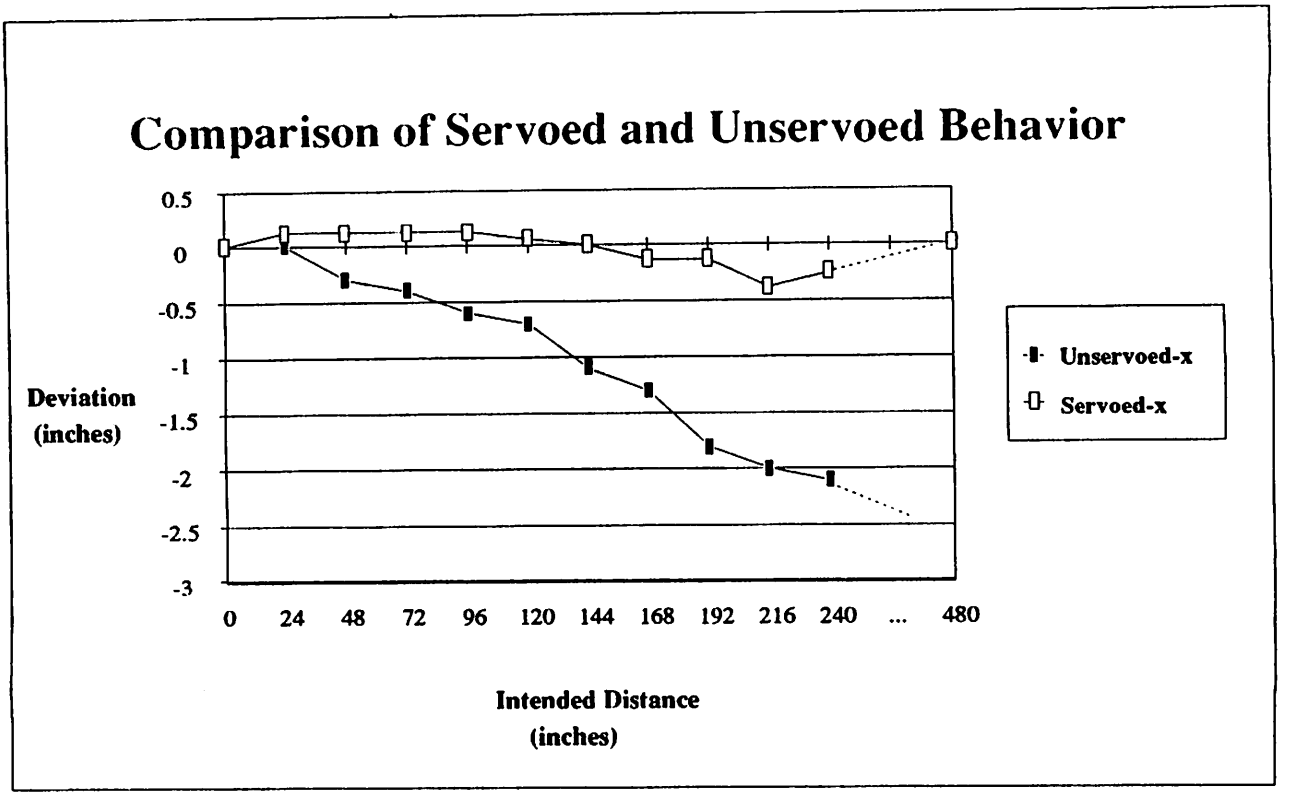


Figure 11: Graph of results tabulated in Table 1.

6 Recognizing Milestones: Plan-Level Perceptual Servoing

Action-level perceptual servoing makes it likely that primitive subgoals are converted accurately into the proper actions but it does not provide a measure of whether or not that subgoal has been satisfied. Using only action-level servoing it is still possible that errors could accumulate, resulting in a failure to satisfy the primitive subgoal or a subgoal at some higher level. This accumulated error is controlled in a procedure called plan-level perceptual servoing. When the actions associated with a subgoal are complete the system attempts to verify the perceptual expectations associated with the milestone following that subgoal in its plan sketch. Minor deviations from these expectations trigger adjustments in the plan sketch hierarchy which correct for the accumulated error. Verifying the perceptual expectations, or milestone recognition, is made simpler and more efficient since the accuracy of the action-level perceptual servoing makes it likely that the expectations and reality will nearly agree.

In contrast to the approach developed by Lowe for the SCERPO system [28,29] we have chosen to separate 3D model-to-image matching into two steps: 2D model matching in image space (discussed in Section 6.1) is followed by the recovery of 3D location and orientation (discussed in Section 6.2) using the correspondences established in the first step. These sub-tasks are interdependent, since an object's position relative to the camera in 3D space cannot be computed without first determining a correspondence to image features, while the correct correspondence depends on the object's 2D

appearance and hence its relative position and orientation in space.

There are several reasons for solving as much of the matching problem as possible in 2D image space. Given the fragmentary data often encountered in complex images, the combinatorics involved in establishing correspondences between landmark models and image lines is particularly severe. Beveridge et al [1,3,4] have shown that the determination of the optimal position of an object's (e.g. landmark's) 2D projection with respect to the endpoints of a set of image lines has an analytic solution in two dimensions. It is highly doubtful that the related 3D problem has an analytic solution for determining model positions that minimize point-to-line and point-to-plane distances. Solving the combinatorial optimization problems of model-to-image correspondence in 3D would then involve the repeated application of a complex spatial optimization problem during the search for the optimal correspondence.

6.1 2D Model Matching

The first step in recovery of the robot position and orientation is matching the 2D projection of landmarks to the line data extracted from the image. Given that matches between landmark models and image data will seldom if ever be perfect, the emphasis must be on determining the 'best' of the imperfect matches. Hence matching is naturally posed in terms of optimization over the possible matches. By establishing an objective measure of match quality, the problem becomes one of determining the correspondence between model elements and data line segments for which the measure is a minimum. The objective function may be task dependent and could incorporate expectations from the environment; this is an open research issue.

The correspondence problem is combinatorial and generally involves mapping one landmark line to many data lines. A second optimization problem is implicit in the correspondence problem. In order to measure the quality of a given model-to-data correspondence, the best 2D position of the model with respect to the data must be determined. This we call the ***spatial fitting*** problem. Hence, a match involves both a model-data correspondence and an associated best-fit position given the correspondence.

There are three components of our approach to the 2D correspondence problem:

1. A search space of possible model-data correspondences.

2. A method for determining the optimal spatial fit for a given 2D correspondence.

3. An objective match measure for evaluating the spatial fit for a given 2D correspondence.

6.1.1 The Search Space

The search space may be viewed as a graph in which the nodes correspond to a particular model-to-data correspondence (a state) and the arcs to state transitions between the correspondences. The state transition operator adds/deletes k data lines from the model-data line correspondence at the current state to create a new state; although values of k 1 are desirable, the computation increases as $O(N^k)$ where N is the number of possible pairings of model and data lines. In practice, $k = 1$ sometimes leads to local minima (non-optimal model positions with respect to the data) in the evaluation function because it does not allow a given correspondence for a single model line to be replaced by another in a single step. (Note that replacement of a model-data pairing involves a deletion and an addition of another model-data line pair.)

Figure 12 shows a simple 2D model matching problem. The table in Figure 12d shows the state transitions during the search for the final match (Figure 12c); each row corresponds to a state and the transition from one state to the next is the addition or deletion of exactly one model to data line correspondence. In the initial state, model line C is associated with data line 8 and model line D is associated with data line 7.

Determining a good starting correspondence and the selection of the k lines to add/delete are open research issues which are currently being examined [1,4]. Finding characteristic substructures (key features) and using them to initially position the model is one possibility for finding the initial correspondence that is being explored. The second involves using a generalized Hough transform and selected key features to determine candidate translation and rotation parameters which are then used to initially position the model.

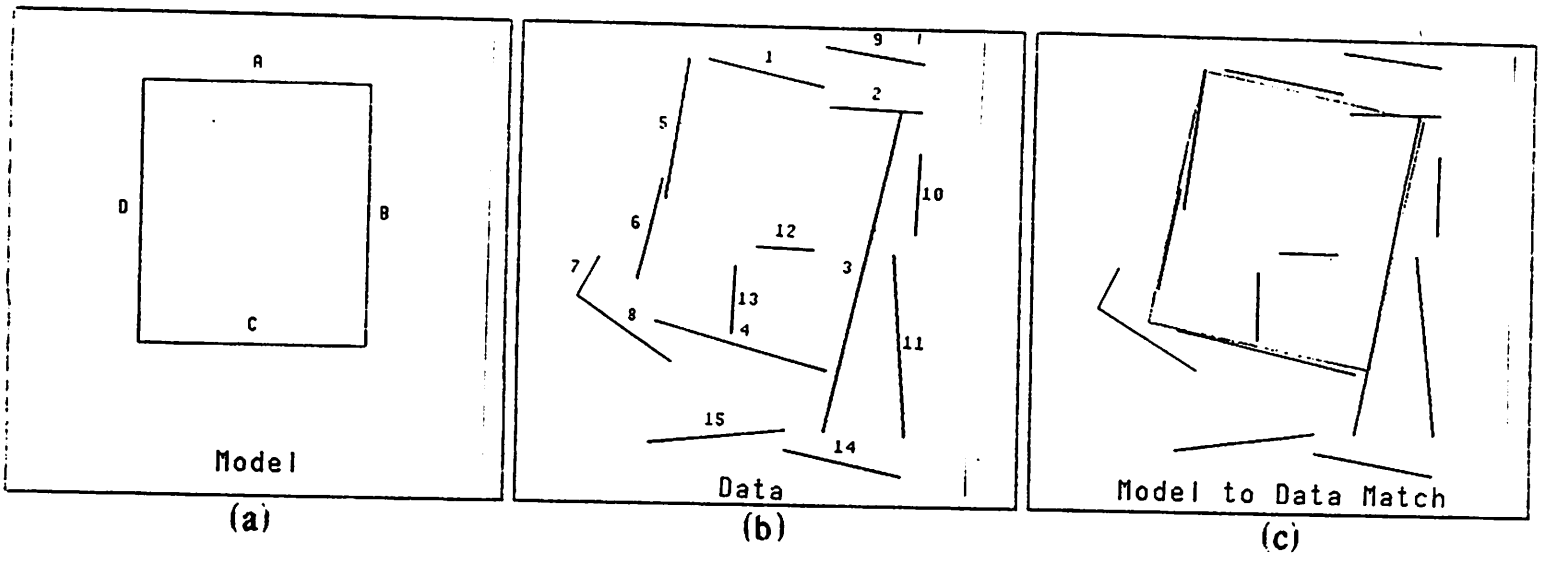
6.1.2 Optimizing the Spatial Fit of the Model

For a given model-data line correspondence (i.e. a given state in the search space), computation of the spatial fit error should be done with the model optimally placed relative to the corresponding data lines. Consequently, the spatial fit must be optimized separately for every state visited in the search space by finding the rotation **R**, translation **T**, and scale factor s which minimizes the spatial fit of the model to the data lines.

A quadratic error model is used in which the fit error E_{fit} is given by:

$$E_{fit} = \Sigma_i (l_i(\mathbf{n}_i \bullet (\mathbf{R}_{(pi)} + \mathbf{T}) - s\rho_i)^2$$

Each term in the sum denotes the squared perpendicular distance from data line endpoint p_i



PAIRINGS, MODEL TO DATA LINE SEGMENTS

	A			B			C					D				
STEPS	1	2	9	3	10	11	4	8	12	14	15	5	6	7	13	EVALUATION
Search Path from Partial Match																
Initial								√						√		17.44
1	√							√						√		15.31
2	√			√				√						√		12.47
3	√			√			√	√						√		10.52
4	√			√			√							√		7.47
5	√			√			√						√	√		5.26
6	√			√			√						√			2.67
7	√			√			√					√	√			1.88
final	√	√		√			√					√	√			1.61

(d)

Figure 12: A simple 2D matching problem. (a-c) The model, line data, and the final match of the model to the data. (d) A trace of the local search process leading to the match in (c). A check designates a correspondence between a model line and data line. The left hand column indicates the state of the search and the right hand column shows the error associated with this state (see Section 6.1.3.)

to the corresponding infinitely extended model line L_i. ρ_i is the distance from the origin to data line L_i in the direction of the unit normal $\mathbf{n}_i$. The contribution of each data line is weighted by its length l_i.

Determining the values of **R**, **T**, and s which minimize E_{fit} requires the solution of only a second order equation; see Beveridge et al [4] for a derivation. The lack of an equivalent solution in 3D is one reason we argue for solving as much of the matching problem in 2D as possible.

6.1.3 The Objective Function

The objective function defines the match error between the optimally placed model and the corresponding data lines. It determines the quality of a match at each state in the search space and is used to guide the search toward a (possibly local) minimum. The objective function is based on the observation that a good match is one in which every model line is consistently accounted for by the data. A match can fail to meet this criteria in two ways: (1) data lines may be displaced from the model line, and (2) portions of the model may be missing in the data. Consequently, the objective function has two terms corresponding to these two conditions:

$$E_{match} = \Sigma(E_{fit}(L) + \alpha E_{omission}(L)) = E_{fit} + \alpha \Sigma E_{omission}(L)$$

E_{fit} is the spatial error computed earlier, while $E_{omission}$ is defined to be a non-linear function of the percent P of the model line unaccounted for by the data. Currently, we use

$$E_{omission} = e^{\beta P}$$

The two terms could be differentially weighted, perhaps on the basis of model line length. The two weights α and β were selected on the basis of extensive experiments with the 2D model matching system. In the results reported below, they were constant at $\alpha = .5$ and $\beta = 4$ throughout all the experiments. In Figure 12(d), the right hand column shows the value of the objective function for each state in the search.

6.1.4 Experimental Results

Figure 13 shows typical results from the 2D model matching system applied to a single frame of a six frame sequence obtained from the robot during an outdoor navigation experiment. Figure 13a is the original image and 13b is the projection (using the assumed position of the vehicle) of six landmarks obtained from the geometric model of the environment.

Assuming the projections to be approximately correct, data line segments within 30 pixels and 0.3 radians of the projected model line segments voted in a generalized Hough transform; the largest peak was used to initially position the model. From this position, a restricted space of correspondences was formed from pairs of model and data line segments within 3 pixels and .3 radians of each other. Finally, the best match in this restricted space was found using a first- improvement local search strategy with $k = 1$. Figure 13c shows the data line segments corresponding to the best matches to the model lines and Figure 13d shows the matched 2D model lines laid over the original image. Additional matching results can be found in Beveridge, et al [1,4].

6.2 Recovering the 3D Location and Orientation of the Vehicle

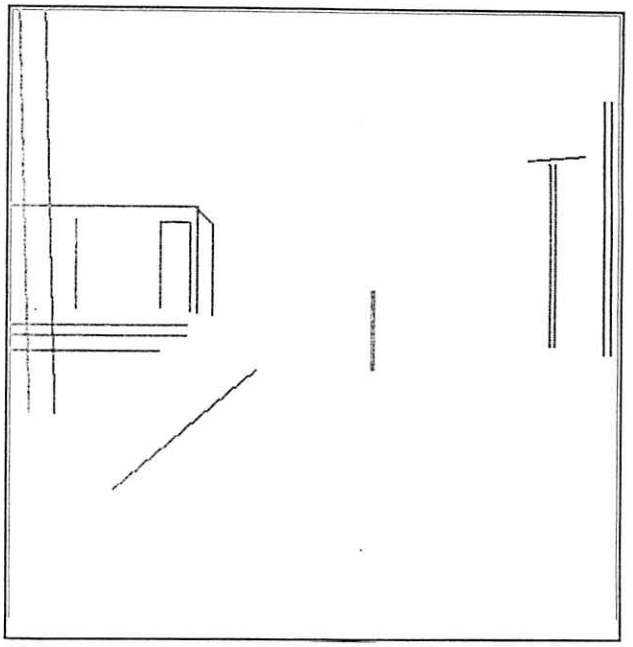
Given the 2D model-data line correspondences between the 3D landmark model lines and 2D image lines, the goal is to find the camera (or robot) 3D rotation **R** and translation **T** which map the world coordinate system to the camera coordinate system under perspective projection. This problem, under various names and guises, has been addressed by several researchers, e.g. see [16]; [20,26,36]; most of the techniques assume point (e.g. line endpoint) correspondences, are iterative in nature, generally minimize the sum of squares of an error estimate, and typically require an initial estimate for **R** and **T**. As noted in Section 3, intrinsic camera parameters, such as focal length, field of view, center of the image, size of image, etc. are assumed to be known [19,23,24,25].

6.2.1 Rotation and Translation Constraints

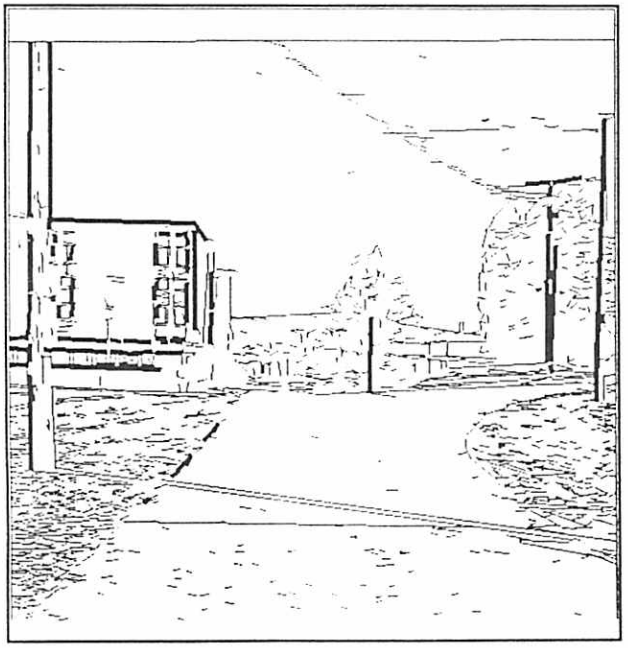
Liu et al [27] develop a solution to the "camera location determination" problem which works for both point and line data. The constraints on **R** and **T** are derived from the observation that the 3D lines in the camera coordinate system must lie on the projection plane formed by the corresponding image line and the optical center. From Figure 14, the rotation constraint can be expressed as $\mathbf{n}_i \bullet \mathbf{R}(d_i) = 0$, where $\mathbf{n}_i$ is the unit normal to the projection plane and d_i is the direction vector of the 3D line. Similarly, the translation constraint captures the observation that the vector formed from the origin through any point on the 3D line (p_i) must be perpendicular to the normal of the projection plane, or $\mathbf{n}_i \bullet (\mathbf{R}(p_i) + \mathbf{T}) = 0$. Liu et. al. used these constraints to solve first for rotation and then for translation using the rotation result; we call this aglorithm **R** then **T**.

Unfortunately, small errors in computing the rotation are amplified to large errors in translation, as is also the case in the recovery of general motion parameters [12]. The technique developed by

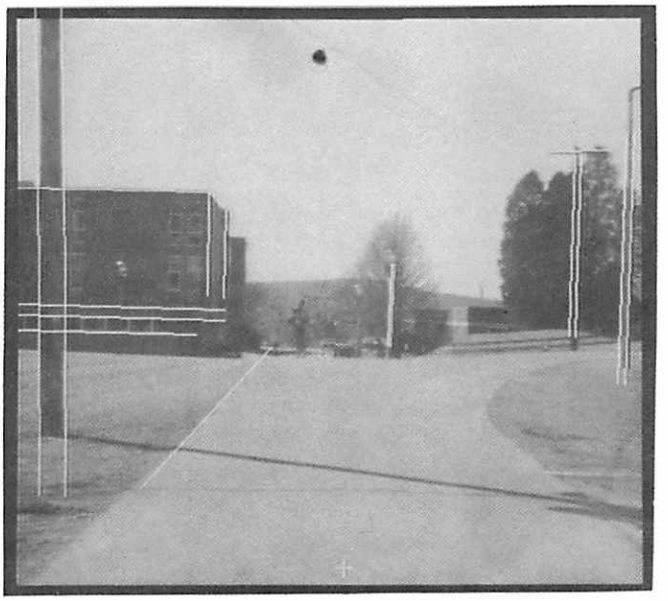
a



b



c



d

Figure 13: Results from 2D Model Matching. (a) One 512 x 512 frame from a six frame sequence acquired by the robot. (b) The six navigational landmarks projected to the image. (c) Data line segments matching the landmark lines. (d) A typical 2D mismatching error: the street light and telephone pole have been matched to the telephone pole and the two lamposts in the background have been matched to erroneous data.

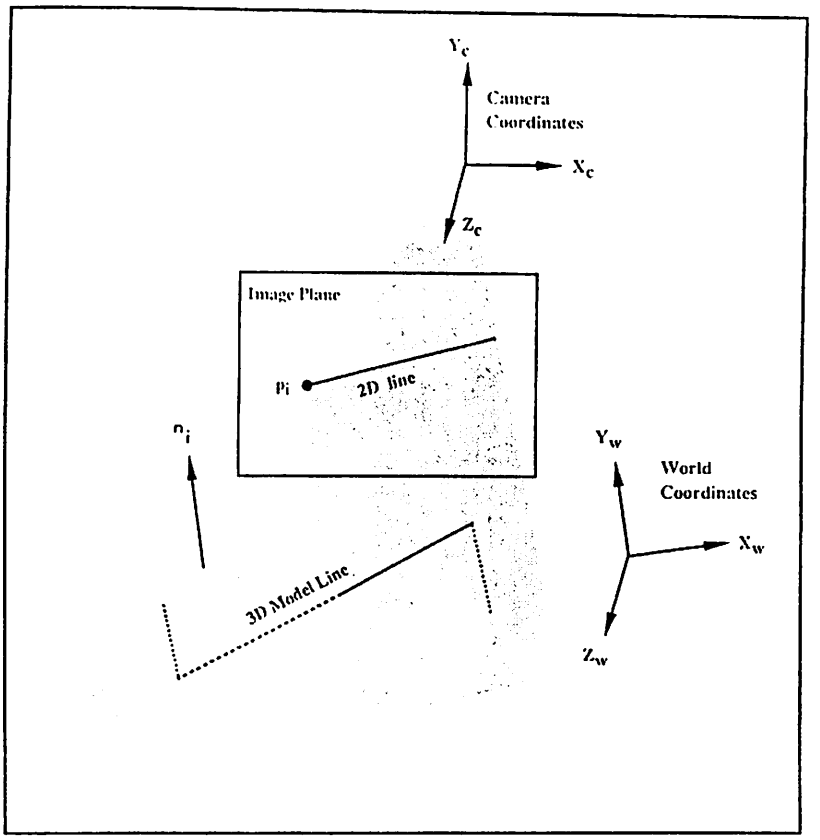


Figure 14: Geometric interpretation of the rotation and translation constraints. Ideally for all points the vector formed from the optical center to a point on the 3D line must be perpendicular to the normal of the projection plane. By minimizing the sum of squares of the perpendicular distances, the optimal R and T can be recovered.

Kumar [23,24] to solve for the rotation and translation parameters differs from that of Liu et al in two significant ways. First, the second of the constraints is used to solve simultaneously for **R** and **T**; this algorithm is called **R and T**. Second, the nonlinear technique used to solve for rotation and translation is adapted from Horn [20] and provides much better convergence properties than does Liu et al's solution method based on Euler angles. The **R and T** algorithm also exhibits much better immunity to noise than does **R then T**.

6.2.2 Solution Technique

Ideally, we would like to find the rotation **R** and translation **T** for which the constraint is satisfied for each line. In the presence of noise, however, this will generally not be possible. Consequently, the problem is cast as one of optimization with an objective function given by:

$$E_1 = \sigma_i(\mathbf{n_i} \bullet (\mathbf{R}(p_i) + \mathbf{T}))^2$$

The objective function E_1 is minimized by finding the **R** and **T** such that the sum of the square of the perpendicular distances from the end-points of the 3D model lines to their corresponding

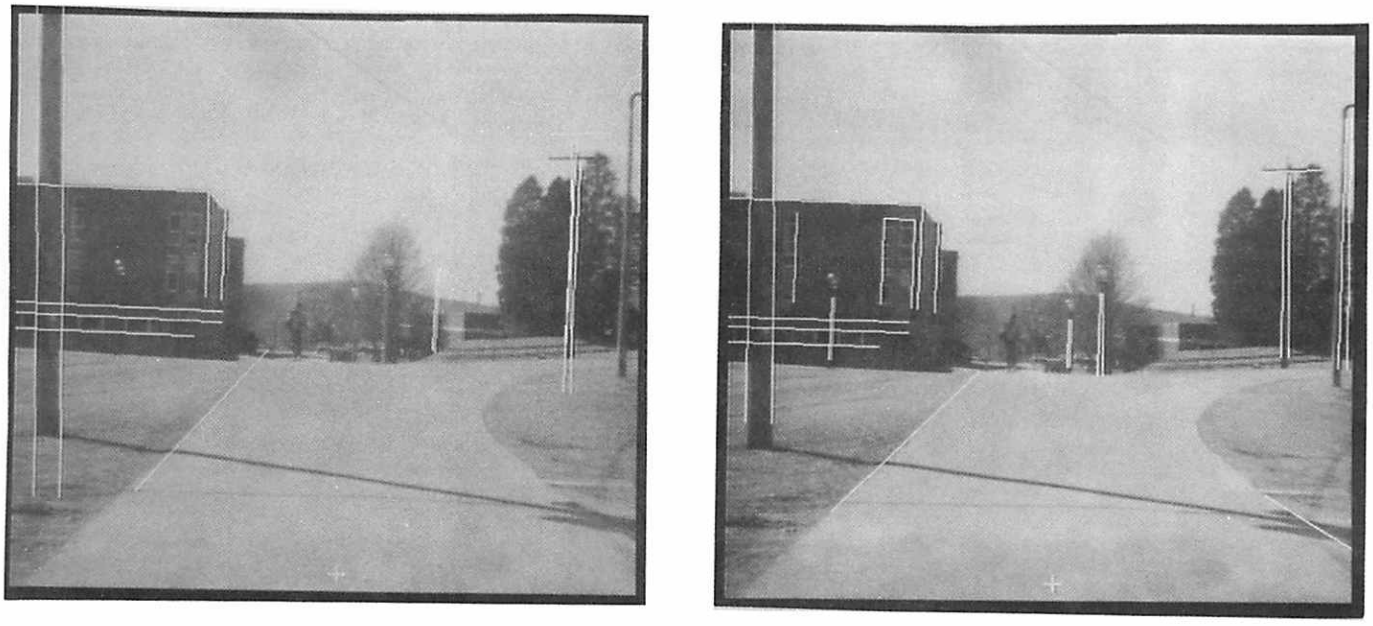


Figure 15: Results from R and T. The white lines are the 3D landmark lines reprojected onto the image plane after recovery of the vehicle orientation using the landmark-image data correspondences shown in Figure 12.

projection planes (formed using the 2D model lines in the image plane) is made as small as possible.

Horn's nonlinear iterative technique requires an initial estimate for both **R** and **T**. The initial estimate is obtained from the vehicle's location and orientation in the previous frame composed with the estimated motion of the vehicle between frames (which undoubtedly has some error). The error term is linearized around the current estimate for **R** and **T** by differentiating E_1 and adding an incremental rotation and translation vector to the current estimates of **R** and **T**. The resulting iterative descent algorithm is terminated when the difference between two successive iterations is less than a threshold.

E_1 gives higher weighting to endpoints of 3D lines which are further away from the camera. In order to give equal weighting to all endpoints, the objective function $E_2 = \sigma_i[(n_i \cdot (\mathbf{R}(p_i) + \mathbf{T}))^2/|(\mathbf{R}(p_i) + \mathbf{T})|]$ could be used. Unfortunately, E_2 is a rational function and is more difficult to optimize than E_1; see Kumar et al [24] for details on the optimization technique. We have found experimentally that E_1 works as well as E_2 and hence E_1 is used.

6.2.3 Experimental Results

Table 2 gives comparative results for **R then T** versus **R and T**. The table shows the absolute average error (in feet) of the computed translation over 100 data samples as a function of noise in the data. The synthetic data were created using the outdoor model with zero mean and uniform random noise added to the ρ and θ of the 2D projected image lines. Some number of 2D lines N(where N=5,10,30) were selected by hand and the pose recovery algorithms were applied only to those lines. In Tables 2 and 3, z is the direction of vehicle travel, x is perpendicular to the ground plane, and y is lateral displacement; all measurements are in feet. Only the translation errors are shown; similar results were obtained for rotation errors [24]. The superiority of the **R and T** algorithm over the **R and T** algorithm is obvious from the table.

Applying the **R and T** algorithm to the 2D matching results shown in Figure 13d, the errors (in feet) for the position of the robot (x,y,z) are (.1, 06, .03). These errors were determined by hand measurement of the robot's position during the experiment; additional results for the other frames of this and similar sequences are given in [23,24].

Table 2. COMPARISON OF TRANSLATIONAL ERRORS
(Error measurements are in feet)

No.	θ	ρ	R then T			R and T		
Lines	deg.	pixels	ΔT_X	ΔT_Y	ΔT_Z	ΔT_X	ΔT_Y	ΔT_Z
5	1.0	1.0	11.44	13.96	51.16	0.21	2.03	1.16
5	5.0	5.0	32.69	39.85	149.40	1.08	10.14	6.20
10	1.0	1.0	9.24	8.83	8.84	0.02	1.73	0.08
10	5.0	5.0	40.83	40.03	38.65	0.18	6.33	0.48
30	1.0	1.0	0.40	1.01	0.36	0.06	0.48	0.06
30	5.0	5.0	2.09	5.05	1.82	0.32	2.39	0.32

6.2.4 Extension of R and T to Data with Outliers

Figure 15a shows a result of 2D model matching which contains typical mismatching problems, although these particular errors were created by hand by perturbing some of the correct 2D matches shown in Figure 13d. In this figure, two lampost models are matched to the same image data, and both the telephone pole model and the street light landmarks have matched to the street light line data. There are 17 3D model lines in this figure, 5 of which are outliers (i.e. grossly incorrect

matches). Application of the **R and T** algorithm results in large errors in the computed position of the vehicle because of the reliance on least-square fits of the data (see line 1 in Table 3). In order to compensate for this problem, a third version of the algorithm, adapted from [31], was developed which minimizes the median error (called **Med R and T**):

$$E_m = Median_i(\mathbf{n_i} \bullet (\mathbf{R}(p_i) + \mathbf{T}))^2$$

Since the median is not a differentiable function, E_m must be minimized by a combinatorial algorithm. Two versions of the algorithm were developed in which the combinatorics were applied to the set of n input lines or to the set of m landmarks (each landmark is a set of lines). The algorithms are capable of correctly computing the correct location and orientation using data containing less than 50% outliers.

Briefly, the algorithms can be summarized as:

- Step 1. Find all 3 line (line version) or 3 landmark (landmark version) subsets of the input data.
- Step 2. For each subset, determine the pose by applying **R and T**; estimate the residual error for all n lines given this pose and find the median-square-error (i.e. the error value associated with the single line or landmark having the median error in the subset).
- Step 3. Select the pose which gives the minimum median-square-error. Remove lines as outliers whose square of the residual error for that pose is greater than a certain threshold.
- Step 4. Run algorithm **R and T** on the remaining lines and return the estimated translation and rotation matrices as the final values of **R and T**. This allows all non-outliers to participate in the optimized solution.

On the basis of time complexity, the landmark version of the algorithm is more efficient than the line version. If the task domain allows a probabilistic determination of the optimal solution (i.e. a probability P that the optimal solution has been found), then a much smaller percentage of the 3 line/3 landmark subsets need to be tested [24].

Figure 15b shows the results of **Med R and T** using the 2D model matching data in Figure 15a as input; the outliers have been correctly identified and the pose determined correctly. Table 3 shows the errors in the translation component of the vehicle position for this frame (Outdoor Frame 1) and several others for **Med R and T** and **R and T**, respectively. Figure 16a-c graphically

compares the results of **Med R and T** and **R and T** for one frame (Indoor Frame 1 in Table 3) of an indoor image sequence. The effect of the outliers on **R and T** is clearly shown in Figure 16b. All results shown are from the landmark version of the algorithm.

Table 3. COMPARISON OF TRANSLATIONAL ERROR
(all measurement are in feet)

Algorithm:			Med R and T			R and T—		
Frame No.	Num. Lines	Num Outliers	ΔT_X	ΔT_Y	ΔT_Z	ΔT_X	ΔT_Y	ΔT_Z
Outdoor 1	17	5	0.80	-0.23	0.20	2.90	0.15	1.10
Outdoor 3	17	3	-1.10	0.14	0.10	34.74	7.34	2.3
Indoor 1	19	8	0.20	0.07	0.10	5.75	0.40	0.60
Indoor 7	46	8	0.20	0.10	0.10	0.30	0.10	0.40

7 Discussion of Experimental Results

The experimental results discussed in sections 5 and 6 are quite encouraging. Each of the perceptual algorithms that has been employed - perceptual servoing, 2D model matching, and pose refinement - seems to have significant potential for robustness when examined individually. However, they effectively complement each other when combined into a system. They either constrain and limit the processing of the next stage, or detect and recover from errors that may have occurred in the previous stage. Thus, we believe that the perceptual algorithms, as they are integrated with the planning and execution modules in our robot navigation system, has true potential for robustness in complex natural domains.

The system for reactive goal refinement and how it is tied to perceptual reasoning will be a subject of ongoing research. Figure 17 is an example result of the response of the system to a goal that is typical of the navigation problems the system was designed to address; plans such as this are being successfully executed in our experimental environment. In this particular experiment, the vehicle stopped within 1 inch of the final goal. A long-term goal of our project is to be able to run this system in real time, i.e. at frame rate while the robot is moving. This goal seems achievable given the form of our system and associated hardware in development. The system has been designed specfically for perceptual strategies that bound the required computation. The environmental model is used to focus the navigational plan on selected landmarks that are

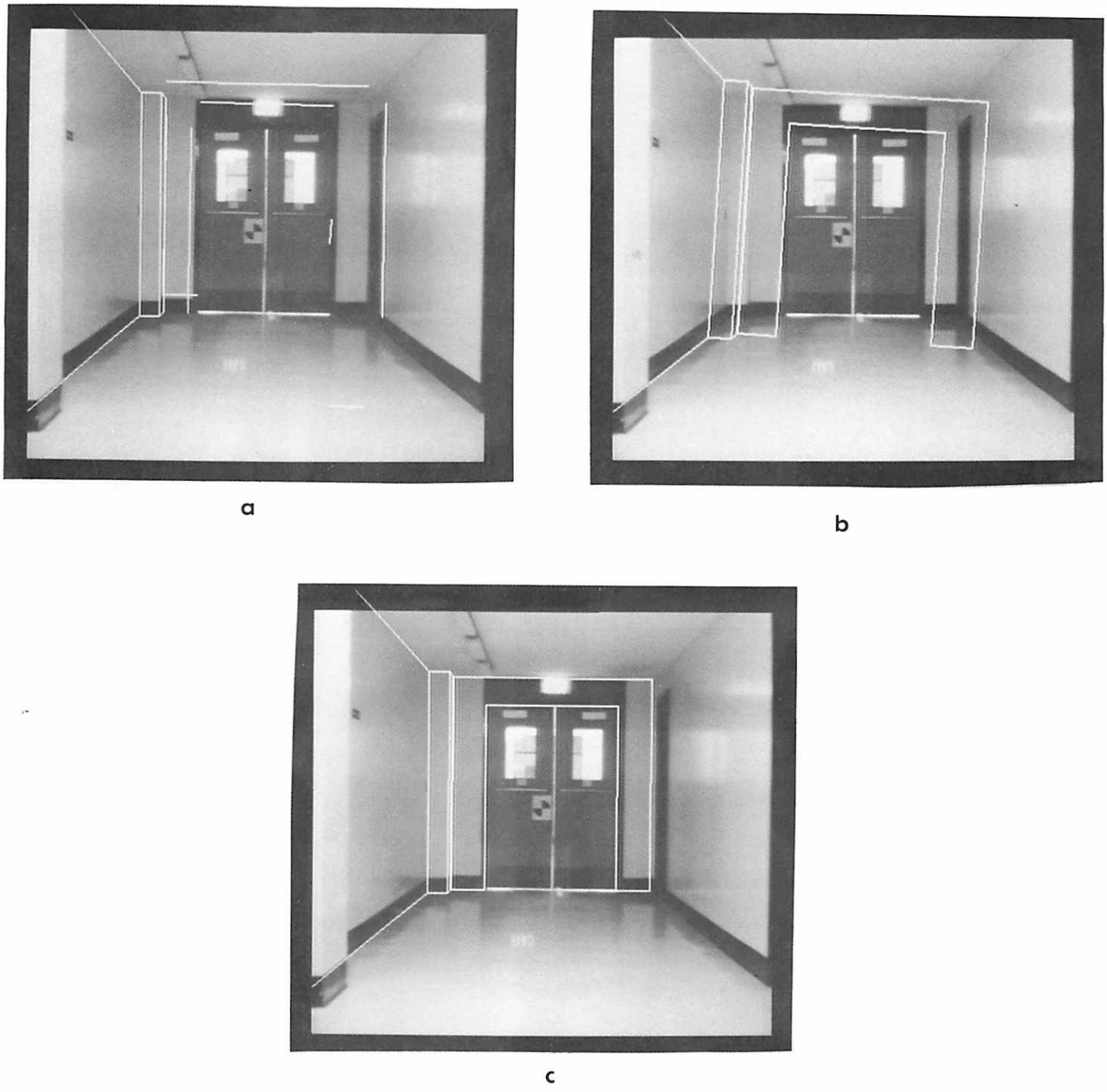


Figure 16: Results from Med R and T. (a-c) One frame from an indoor sequence (Indoor Frame 1 in Table 3) comparing R and T and Med R and T. (a) Landmark matches after 2D matching with 8 of 19 lines as outliers; this is the input data to the 3D pose recovery algorithm. (b) Projected lines after estimation of position and orientation using R and T; (c) same as (b) but using Med R and T. The effect of outliers in least squares error estimation is clearly visible in (b).

distinctive; unnecessary planning is avoided by partial incremental planning with milestones as "reality checks;" and the perceptual servoing during motor action is designed to avoid losing the landmarks (and robot pose) at a modest and efficient computational cost.

A sampling of the time spent in computation should provide a feel for why we are optimistic about real-time computing rates. The experiment described in Section 5, which was run using a combination of Vax CommonLisp and C, took 20 minutes on a time-shared Vax11- 750. About 30% of this time was spent computing. The planning system, written in Vax CommonLisp , takes about 5 minutes to complete the entire set of plan sketches when run on a Vax11-750 (this set consists of the 3 sketches shown in Figures 17b,c,g and a total of 21 milestones; not all milestones are shown). Model matching is the single most computationally complex operation in the system. Even written in C, it is possible for the program to run for hours if the search is not focussed. Our experience so far has been that we have been able to keep this time down to 1-5 minutes on an 8-node Sequent multiprocessor by keeping tight bounds on the action execution (and consequently the size of the search space). Pose refinement, written in C, typically completes its task in less than 30 seconds on a Sequent. All of these times include I/O for data display purposes. The UMass Image Understanding Architecuture (IUA) [35] will provide enormous computational speedup when it arrives in the coming year.

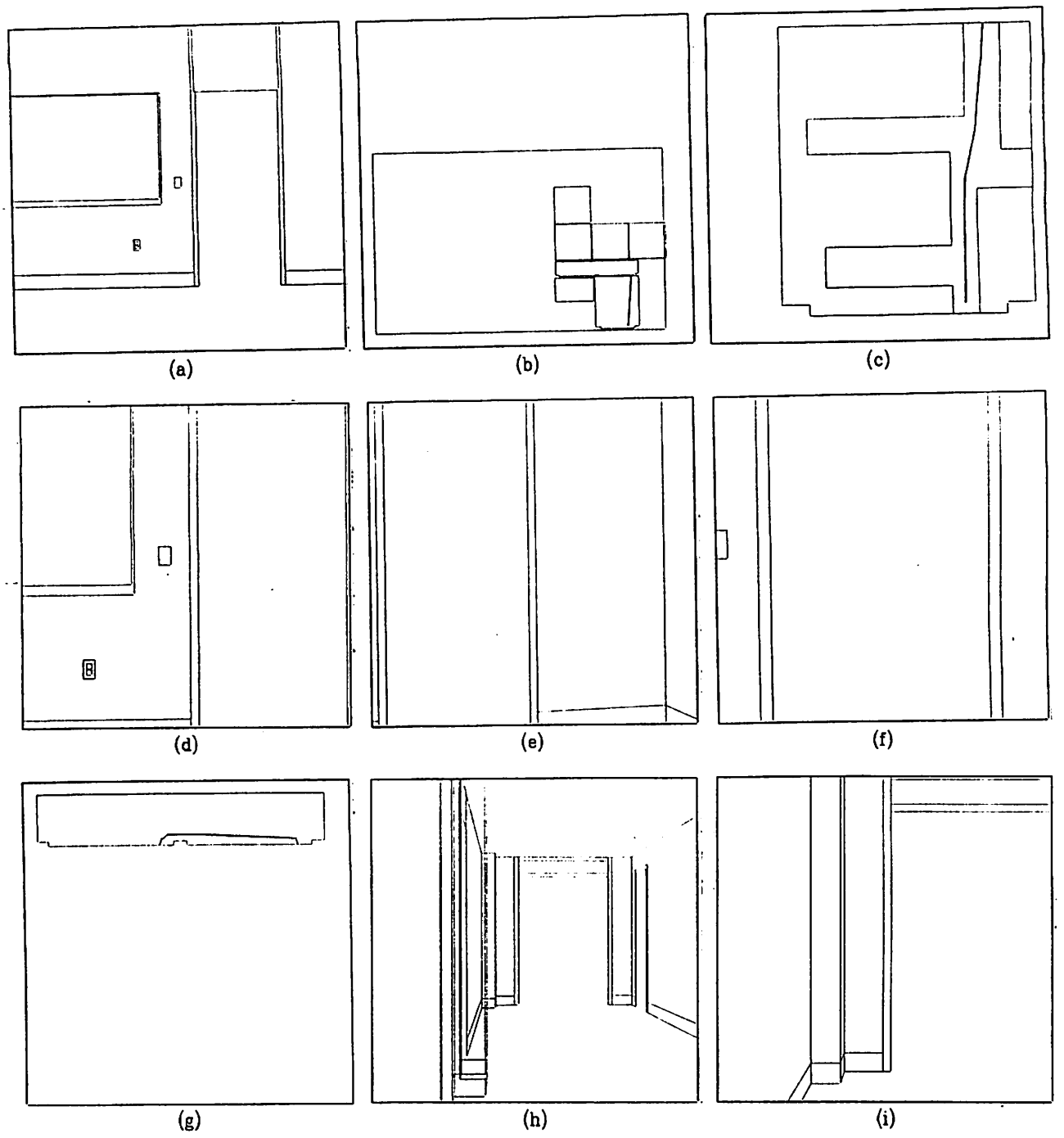


Figure 17: Panels (a)-(i) are displays output by Harvey's reasoning system, given the goal (ptrans (robot-laboratory (16,2,0) (allens-office (13,12,0)). A precondition milestone is generated, (a) as part of a high level plan sketch. (b) (Milestones actually used by the system are only a subset of what is displayed here. These displays, which are created by the same process, are more complete for illustration purposes.) A more detailed analysis of the first goal in (b) makes it clear that plan sketch refinement is necessary, resulting in (c), a plan sketch which avoids the known obstacles in the robot-laboratory. Panels (d)-(f) show sample milestones generated for the path from the starting point to the door from the robot-laboratory to the hallway. Panel (f) is a model projection of the door-way, indicating that the robot is ready to enter the hall. Entry into the hall triggers a plan sketch refinement to negotiate the hallway. Finally (h) and (i) are hallway milestones, (i) is the milestone indicating that Harvey is ready to turn to enter the door- way into the office.

8 Summary and Conclusions

The work presented in this paper is the current status of a long-term research effort on perceptual systems for navigation of autonomous mobile robots. The focus of the research is on environmental modeling, planning, plan monitoring, and verification via model-based vision. These four components are tightly coupled in a system which provides the flexibility and extensibility required for an experimental testbed for robot navigation.

Because the vagaries of the physical world affect plan execution in unknown ways, plans cannot simply be blindly executed, no matter how carefully constructed. Each step of a plan must be carefully monitored and compared to expectations. The system accomplishes this by defining milestones associated with each planned action. The milestones act as preconditions for subsequent plan steps; the next step cannot be executed unless the milestone is satisfied. This assures a correspondence between the environmental model and the assumed position of the robot relative to the model and the actual position of the robot in the physical world. Failure to satisfy a milestone causes replanning to take place. Interweaving perception, planning, and action in this way makes specific what task is expected of perception and provides a means for focusing available knowledge on local goals.

A unique feature of the model matching component is the separation of the process of positional updating into two steps: 2D matching followed by 3D pose refinement. The robustness of this technique and its computational efficacy over many experiments in multiple domains is currently being explored.

More accurate execution of motor actions should allow the 2D model matching and 3D pose refinement process to be applied far more effectively before significant error accumulates. Expected model-data line correspondences will typically be constrained to smaller areas in the image, so many candidate matches can be ignored, allowing significant reduction in the search space of data lines. Since the correspondence problem in 2D model matching is combinatorial, this will greatly reduces the required computation, and of course avoid some of the potential errors from ambiguity.

Experimental results from the system are quite encouraging, although a number of issues remain to be explored. The partial model of the environment is fairly rich with accurate landmarks, which is perhaps an unrealistic assumption for an autonomous robot. It remains to be seen how the requirement of accurate landmarks can be relaxed while maintaining the idea of perceptual servoing. Incorporating the type of reasoning demonstrated by the Schema system [11] might allow the vehicle to execute instructions like "....continue down North Pleasant street past the Graduate

Research Center, then turn left at the corner and..." Ultimately, such capabilites will depend on use of specific landmarks, as well as recognition of instances of general object classes, and further generalization of the capabilities developed here for determination of relative position to these entities.

Finally, navigation is an extremely computationally demanding task, yet real-time performance is crucial for a mobile automaton whose survival may depend upon reaching critical decisions in a short period of time. An ongoing aspect of the work reported here is exploration of means by which the navigation task may be distributed over suitably configured parallel architectures. Two complementary lines of research are currently underway, utilizing a Sequent Symmetry multiprocessor system and the University of Massachusetts Image Understanding Architecture [35].

9 Acknowledgements

The authors wish to acknowledge the many members of the VISIONS research group who have, knowingly or unknowingly, contributed to this effort. In particular, special thanks to Jim Burrill for managing to keep the system software intact through multiple changes, to Val Cohen for keeping Harvey in such good shape, to Dave Ehrenberg for patiently running many of the experiments reported here, and to Laurie Waskiewicz for perservering through multiple drafts of this paper.

10 References

REFERENCES

[1] J.R. Beveridge, R. Weiss, and E.M. Riseman, "Combinatorial Optimization Applied to Variable Scale 2D Model Matching," *10th International Conference on Pattern Recognition*, Atlantic City, NJ, June 1990, to appear.

[2] J. R. Beveridge, J. Griffith, R. Kohler, A. Hanson and E. Riseman, "Segmenting Images Using Localized Histograms and Region Merging," *International Journal of Computer Vision, Vol. 2(3)*, pp. 311-347, 1989.

[3] J. R. Beveridge and R. Weiss, "Optimization of 2-Dimensional Model Matching," *Proc. of DARPA Image Understanding Workshop*, Palo Alto, CA, pp. 815-830, May 1989.

[4] J. R. Beveridge, R. Weiss and E. Riseman, "Optimization of 2-Dimensional Model Matching Under Rotation, Translation and Scale," Dept. of Computer and Information Science, University of Massachusetts (Amherst), TR 89-57, June, 1989.

[5] R. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robots and Automation*, Vol. RA-2(1), pp. 14-23, 1986.

[6] J. B. Burns, A. Hanson and E. Riseman, "Extracting Straight Lines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8(4), pp. 425-455, 1986.

[7] C. Connolly, "Geometer: A System for Modelling and Algebraic Manipulation," Dept. of Computer and Information Science, University of Massachusetts (Amherst), TR89-99, September 1989.

[8] C. Connolly, D. Kapur, J. Mundy and R. Weiss, "Geometer: A System for Modelling and Algebraic Manipulation," *Proc. of DARPA Image Understanding Workshop*, Palo Alto, CA, pp. 797-804, 1989.

[9] E. Dickmans and V. Grafe, "Applications of Dynamic Monocular Machine Vision," *Machine Vision and Applications, Vol. 1*, pp. 241 and following. 1988.

[10] E. Dickmans and V. Grafe, "Dynamic Monocular Machine Vision," *Machine Vision and Applications, Vol. 1*, pp. 223-240, 1988.

[11] B. Draper, J. Brolio, R. Collins, A. Hanson and E. Riseman, "The Schema System," *International Journal of Compuer Vision, Vol. 2(3)*, pp. 209-250, 1989.

[12] R. Dutta, R. Manmatha, M. Snyder and E. Riseman, "Issues in Extracting Motion Parameters and Depth from Approximate Translational Motion," *Proc. of IEEE Workshop on Visual Motion*, Irvine, CA. pp. 264-272, March 1989.

[13] C. Fennema, E. Riseman and A. Hanson, "Planning With Perceptual Milesto nes to Control Uncertainty in Robot Navigation," *Proc. of SPIE - International Society for Photographic and Industrial Engineering, Mobile Robots III*, Cambridge, MA, pp 2-18, 1988.

[14] C. Fennema, E. Riseman and A. Hanson, "Planning with Perceptual Milestones to Control Uncertainty in Robot Navigation," *Proc. of AAAI Spring Symposium Series (Working Notes: Robot Navigation)*, Stanford University, pp. 19-23, March 1989.

[15] C. Fennema, A. Hanson, E. Riseman. (1989b). "Towards Autonomous Mobile Robot Navigation," Dept. of Computer and Information Science, University of Massachusetts (Amherst), TR 89-104, October 1989.

[16] S. Ganapathy, "Decomposition of Transformation Matrices for Robot vision , *Proc. of 1st IEEE Conference on Robotics*, pp. 130-139, 1984.

[17] M. Herman, T. Hong, S. Swetz, D. Oskard, and M. Rosol, "Planning and World Modeling for Autonomous Undersea Vehicles", *Proc. Third IEEE International Symposium on Intelligent Control*, Arlington, VA, August 1988.

[18] M. Herman and J. Albus, "Overview of the Mutliple Autonomous Underwater Vehicles (MAUV) Project," *Proc. IEEE International Conference on Robotics and Automation,* Philadelphia, PA, pp. 618-620, April 1988.

[19] B. K. P. Horn, "Robot Vision", Cambridge, MA: MIT Press, 1986.

[20] B. K. P. Horn, "Closed-Form Solution of Absolute Orientation Using Unit Quarternions," *J. Opt. Soc. A.*, Vol. 4, pp. 629-642, 1987.

[21] P. Kahn, L. Kitchen and E. Riseman, "Real-Time Feature Extraction: A Fast Line Finder for Vision-Guided Robot Navigation," Dept. of Computer and Information Science, University of Massachusetts (Amherst), TR 87-57, July 1987.

[22] T. Kanade, C. Thorpe and W. Whittaker, "Autonomous Land Vehicle Project at CMU," *Proc. of ACM Computer Conference*, 1986.

[23] R. Kumar, "Determination of Camera Location and Orientation," *Proc. of DARPA Image Understanding Workshop*, Palo Alto, CA, pp. 870-881, May 1989.

[24] R. Kumar and A. Hanson, "Robust Estimation of Camera Location and Orientation from Noisy Data having Outliers," *Proc. of IEEE Workshop on Interpretation of 3D Scenes*, Austin Texas, pp. 52-60, November 1989. A shorter version may be found in Dept. of Computer and Information Sciences, University of Massachusetts (Amherst), TR89-120, December 1989.

[25] R. Lenz and R. Tsai, "Techniques For Calibration of the Scale Factor and Image Center For High Accuracy 3-D Machine Vision Metrology," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10(5)*, pp. 713-719, 1988.

[26] S. Linnainmaa, H. D. and L. Davis, "Pose Determination of a Three-Dimensional Object Using Triangle Pairs," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10(5)*, pp. 634-647, 1988.

[27] Y. Liu, T. Huang and O. Faugeras, "Determination of Camera Location From 2D and 3D Line and Point Correspondences," *Proc. Computer Vision and Pattern Recognition*, Ann Arbor, MI, pp. 82-88, June 1988.

[28] D. Lowe, "Perceptual Organization and Visual Recognition," Boston, MA, Kluwer Academic Publishers, 1985.

[29] D. Lowe, "The Viewpoint Consistency Constraint," *International Journal of Computer Vision, Vol. 1(1)*, pp. 58 -72, 1987.

[30] J. Lowrie, M. Thomas, K. Gremban and M. Turk, "The Autonomous Land Vehicle (ALV) Preliminary Road-Following Demonstration," *Proc. of Intelligent Robots and Computer Vision* (SPIE 579), D.P. Casasent, Ed., pp. 336-350, 1985.

[31] P. J. Rousseeuw and A. M. Leroy, "Robust Regression and Outlier Detection," New York, NY: John Wiley & Sons, 1987.

[32] R. Schank and R. Abelson, "Scripts Plans Goals and Understanding", Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1987.

[33] C. Thorpe, S. Shafer, and A. Stentz, "An Architecture for Sensor Fusion In A Mobile Robot," *Proc. of IEEE International Conference on Robotics and Automation*, Vol 3, pp. 1622 and following, April 1986.

[34] G. Toscani and O. Faugeras, "Structure from Motion Using the Reconstruction and Reprojection Technique," *Proc. of IEEE Workshop on Computer Vision*, pp. 345-348, 1987.

[35] C. Weems, S. Levitan, A. Hanson, E. Riseman, J. Nash and D. Shu, "The Image Understanding Architecture," *International Journal of Computer Vision, Vol. 2(3)*, pp. 251-282, 1989.

[36] P. Wolf, "Elements of Photogrammetry," New York, McGraw Hill, 1974.