Experiments in Autonomous Navigation

Claude Fennema
Allen R. Hanson

COINS TR90-99

October 1990

Experiments in Autonomous Navigation

Claude Fennema and Allen R. Hanson
Computer and Information Sciences Department
University of Massachusetts
Amherst, MA 01003

Abstract[1]

The University of Massachusetts mobile robot project is investigating the problem of goal-oriented navigation of an autonomous robot vehicle through a partially modeled, unchanging environment which contains no unmodelled obstacles. This simplified environment provides a foundation for research in more complicated domains.

Perception, planning, and execution of actions are integrated into a reactive system which reasons about landmarks that should be perceived at various stages of task execution. Perceptual servoing uses correspondences between image features and expected landmark locations to ensure proper plan execution and to maintain the relationship between the system's internal model and the environment. This paper briefly describes this system and presents experimental results which demonstrate the efficacy of perceptual servoing.

1. Introduction

The UMass Mobile Robot project [FEN90] is investigating the problem of enabling a mobile automaton to navigate intelligently through indoor and outdoor environments. The experimental platform, called "Harvey", is a Denning Mobile Robotics vehicle ultimately intended to navigate through offices, hallways, and university grounds as it carries out commands such as "Fetch the book" or "Take this to Claude". The initial system development efforts and 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.

2. System Overview

The philosophy underlying the system is the use of sensory data in a 'verification' mode. 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, which includes a geometrically specified environmental model as well as the robot's spatial relationship to this environment 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.

Planning in this system is reactive. Harvey does not generate detailed plans. Plans are "sketched" and modified in response to what is perceived as a result of each action. To begin with each task given to the robot is decomposed depth first into a tree of less abstract subgoals as shown in Figure 1. The leaves of this tree form a sequence of subgoals which accomplish the task. This sequence is called a plan sketch because it is only partially developed and generally will change as execution proceeds. This plan sketch is detailed at its beginning and becomes increasingly abstract towards its end. Figure 2 shows pictorially what a tree such as in Figure 1 corresponds to for a typical goal Harvey may encounter. Action begins as soon as the first subgoal in the plan sketch is a primitive. All action is directed by the dynamic planning

[1] This research has been supported by the Defense Advanced Research Projects Agency under RADC contract F30602-87-C-0140 and Army ETL contract DACA76-89-C-0017.

and execution monitor, "plan-and-monitor" [FEN 88, FEN 89] which orchestrates planning, action and perception in a fine grained, interwoven architecture.

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, determining milestones, milestone recognition, determination of location, and execution of primitive motor actions. 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 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.

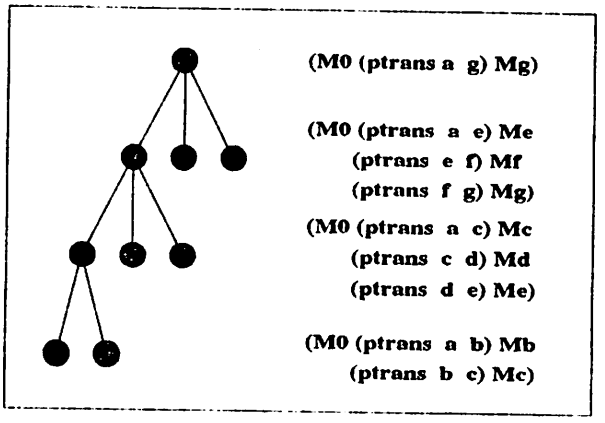

Figure 1. Goals are decomposed in a depth first fashion into less abstract subgoals, forming a tree. Each level of the tree corresponds to a plan sketch which details a subgoal in the level above it. The leaves of the tree correspond to a plan sketch of the complete task, one which is quite detailed at the beginning but becomes quite abstract towards the end.

The navigation system described here differs from existing systems [THO 86, LOW 85, DIC 88a,b, BRO 86, TOS 86, HER 88a,b] in its reliance on an environmental model, high level goals, the use of sensor data in a 'verification' mode, and the close coupling of planning, action, and perception. Many of the techniques utilized by these systems (for road following, for example) could be embedded in Harvey's perceptual servoing mechanisms. Since Harvey assumes there are no unmodelled obstacles in its environment, the obstacle detection and avoidance methods developed for these systems could also be used.

3. The System Model

The system model contains environmental information relevant to the navigation task, including models of the system sensors and actuators, the environment, and the navigation task itself (the plan). The first two components of the system model are discussed briefly below; the task model is described in more detail in Section 4.

3.1 The actuator and sensor model

The actuator and sensor model contains information about the primitive actions the robot is capable of executing directly and parameters of the sensor. The suite of primitive motor actions includes 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. This model is discussed in more detail in Section 4. The sensor model is more complex and is an ongoing research topic. Currently the model consists of: a set of camera parameters including focal length, aspect ratio, image center, and image size; a set mathematical transforms including projection and clipping; and a set of low-level vision modules capable of extracting straight lines, performing simple correlation, and the like.

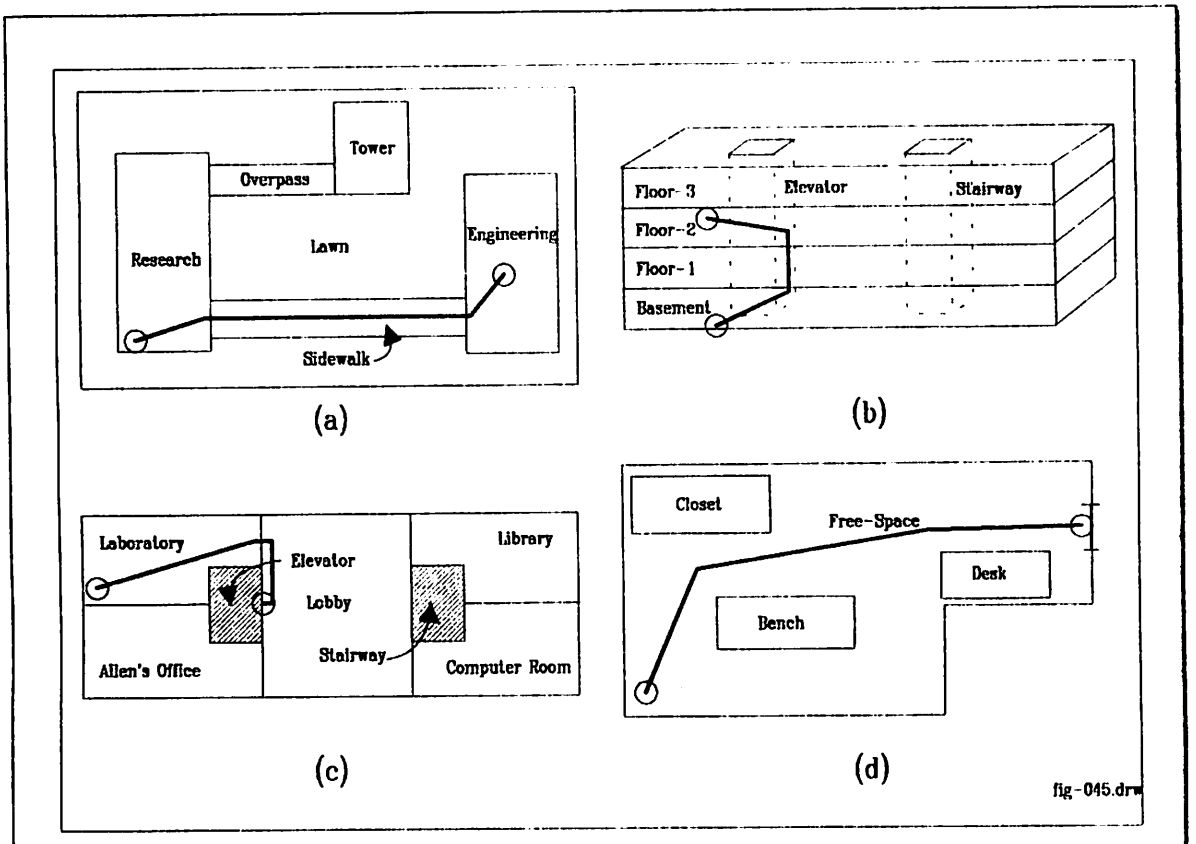

Figure 2. Accomplishing the goal (ptrans robot-lab teds-office) would begin by building a tree such as is shown in figure 1. Here the decomposition is shown pictorially from the most abstract plan sketch (a) to the most specific (d).

3.2 The environmental model

The environmental model specifies the geometric and perceptual structure of the robot's world. Portions of the interior of the University of Massachusetts 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 3 is a projection of a portion of a hallway used in the experiments described in Section 5.

The environment is represented as a graph structure which captures the key topological, geometric, and physical properties of the spatial entities making up the robot's world [FEN 90]. This network describes space in terms of a hierarchical 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 (Figure 4).

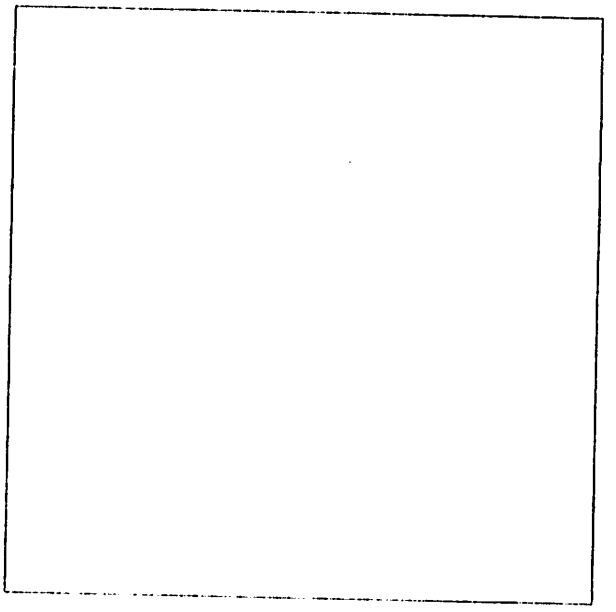

Figure 3. The model of the environment is incomplete, but positionally accurate. All obstacles are modeled but many visual details may be absent.

The geometric properties of each face are represented in terms of lines and points, which are similarly 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 6). 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, for example.

4. Representing Goals and Actions

The task model represents the navigational goal(s) 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 [SCH 76]. The goal to move from one location to another is represented as: (ptrans location-1 location-2)

Milestones are typically specifications of one or more 3D landmarks and their expected location with respect to the robot at the completion of the preceding phase of the plan sketch. Goals and milestones are collected into plan sketches represented as a list of the form:

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

M0 is a precondition milestone which must be satisfied before the plan sketch can be considered meaningful. 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.

As the plan sketch hierarchy is developed, goals are refined in a depth first fashion to more and more detailed levels until the leading subgoal is a primitive subgoal. A primitive subgoal by definition is one which can be directly executed without further subgoal refinement. For the Denning Robot, the primitive

actions are TURN(angle) and MOVE(distance); all primitive subgoals are decomposed into a TURN followed by a MOVE. For example, if the primitive subgoal were

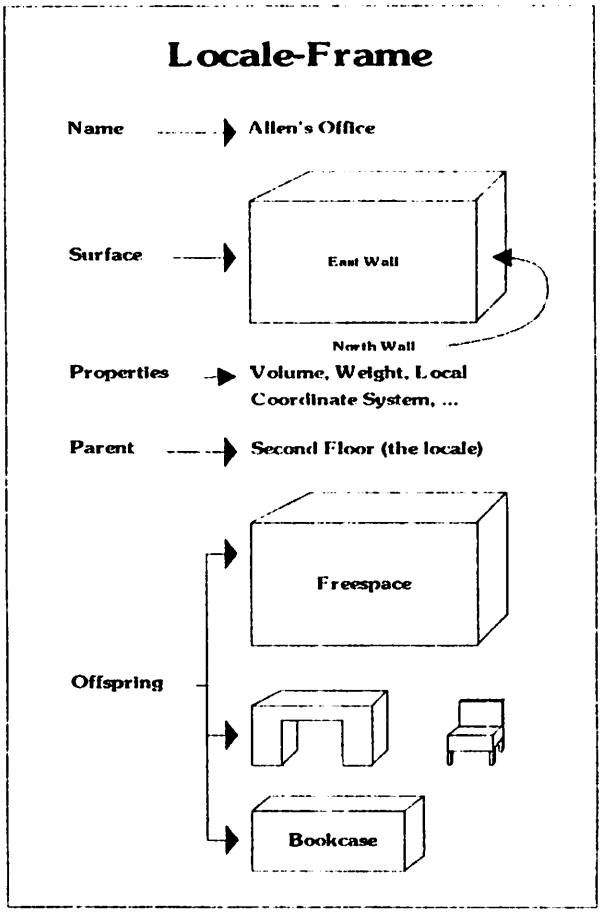

Figure 4. All 3D space entities are represented as locales. These locales are implemented as frames whose slots organize its topological, geometric, and physical properties. Shape and spatial extent are represented by a description of the locale's surfaces in terms of faces (Figure 5).

(ptrans location-1 location-2)

and the path from location-1 to location-2 were unobstructed, then this subgoal could ideally be satisfied by executing the script:

```
(script-ptrans-clearpath (location-1 location-2)
    (turn (- (angle location-1 location-2)) heading)
    (move (distance location-1 location-2)))
```

This script specifies that to get from location-1 to location-2 it is only necessary to perform two actions: first turn the difference between the angle of the line from location-1 to location-2 and the current robot heading, then move the distance between these two locations.

5.0 Perceptual Servoing

The script illustrated in section 4 will not satisfy the subgoal unless the individual actions "turn" and "move" are properly executed. In practice, this is unlikely without sensory feedback because the realities

of the environment and the actuators of the agent make the execution of open loop actions unreliable. As the robot performs "turn", for example, its rubber tires introduce a translational motion or "skittering". Whenever the robot moves forward along an environmental surface, a slippery spot, a unevenness of the surface, or even a bulge in its tire may throw it off course, causing inaccurate execution. To reduce this error the execution of each action is controlled by servoing on prominent visual features in the environment. This servoing using perceptual features is called "action-level perceptual servoing".

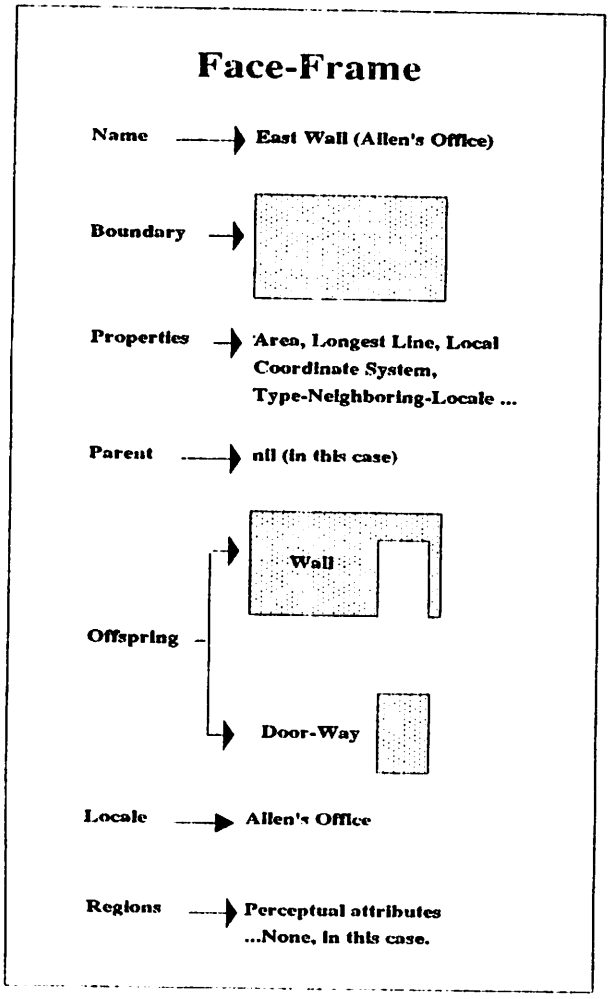

Figure 5. Locale surfaces are currently described in terms of planar faces which, like locales, are represented as frames. Values in the slots organize information such as its type (door-way or barrier) and, if it is a door-way, the name of the locale which shares the doorway.

Action-level perceptual servoing increases the accuracy of each action, but does not relate the result to progress toward the goal, nor does it prevent the accumulation of inaccuracies over a number of such actions. It may be determined, for example, that three steps of distance 1.23 miles are required to accomplish a subgoal. Action level perceptual servoing might execute these actions fairly accurately to, say, 1.2 miles (a 2.4% error). But, after executing these three actions the agent will be .1 mile short of its goal and will very likely not even be able to see the goal from that position, so the subgoal has not been achieved. It is for this reason that milestone were introduced. A better script might be:

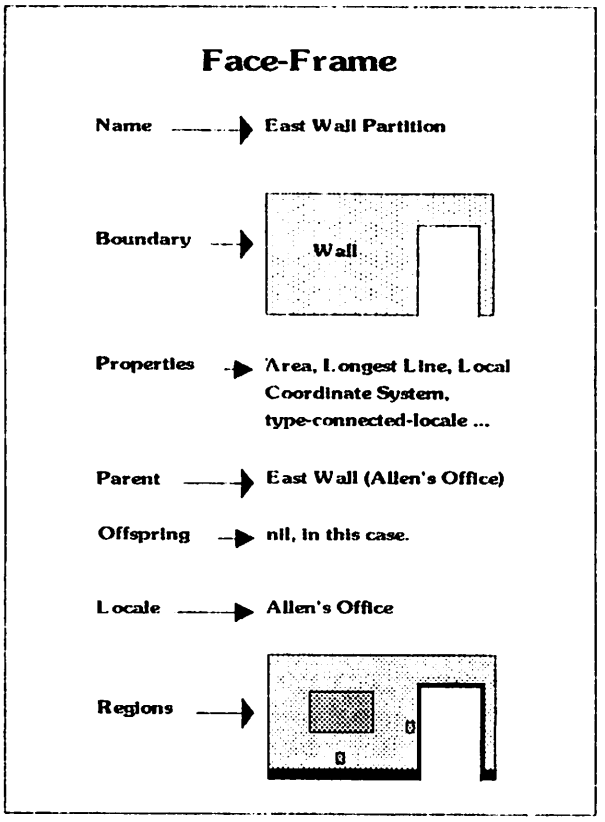

Figure 6. The appearance of each face is described as a collection of regions, each representing an area of the face which has uniform appearance properties (color, texture, etc.)

```
(script-ptrans-clearpath (location-1 location-2)
    (turn (- (angle location-1 location-2)) heading)
    (if (not(recognize-milestone milestone-t))
        (return (list failure last-known-location)))
    (move (distance location-1 location-2))
    (if (not (recognize-milestone milestone-t))
        (return (list failure last-known-location))
        (return (success location-2))))
```

In this script each action is followed by a milestone check. As long as the results of these checks are positive, processing the script continues, otherwise failure is reported. In any case the last known location is returned along with the success/failure report. If the result of processing the script is (success location-2) the subgoal has been achieved. If, however, the result is (failure last-known-location) then the plan sketch hierarchy must be adjusted or redeveloped. If the location last-known-location is sufficiently near location-2, it may be enough to replace the next subgoal to be satisfied, namely (ptrans location-2 location-3), with the new subgoal (ptrans last-known-location location-3) and continue. This procedure is called "plan-level perceptual servoing".

When the distance between the last known location and location-3 is large, however, it will be necessary for the agent to relocate itself and redevelop a major portion of the plan sketch hierarchy. This relocation-redevelopment cycle forms a third level of servoing called "goal-level perceptual servoing".

Navigational tasks are thus accomplished using three nested levels of perceptual servoing: action-level, plan-level and goal-level. The details of goal-level perceptual servoing are outside the scope of this paper. The following discussion describes plan-level and action-level servoing.

5.1 Action-Level and Plan-Level Servoing

Both action-level and plan-level servoing are based on knowledge directed, top down vision. In this project it is assumed the agent has partial knowledge of all obstacles in the environment. In particular it is assumed that the agent has accurate, but not necessarily complete knowledge about the appearance the objects in its environment. A perceptual servoing cycle begins by analyzing what is known about the environment and what should be perceived from the current location of the agent. 3D entities, called landmarks, are selected from the model on the basis of how distinctive they are, and what kind of information they offer the servoing procedure. Once these landmarks are selected their appearance is projected onto the image plane and are matched to data in the image. These matches, along with the knowledge of the 3D locations of the landmarks, are used to compute and make the appropriate corrections to the action or to the plan sketch hierarchy. Both action-level and plan-level servoing use the same landmark selection and matching procedure; they differ only in what they do with the resulting information.

5.2 Selecting Landmarks

In principle a landmark can be any 3D entity: an object, a group of objects, a group of lines [FEN 90. BEV 89, BEV 90], or cluster of surface patches, as described here. Surface patches were chosen because their reflectance patterns can be quite distinctive, making it likely that they can be isolated from other 3D entities in an image with rather simple means, such as correlation. Correlation, properly used, is a strong method for matching such reflectance patterns; it is known to be noise tolerant, and it can be computed quickly on special purpose machinery [DIC 88a,b]. Landmarks are selected on the basis that they will be easy to find using correlation.

Distinctive reflectance patterns occur where regions of differing reflectances meet, namely at boundary segments and vertices. In the following description surface patches defined by vertices are used, but the same principles apply to surface patches defined by line segments or curves. Selecting landmarks from the model, then, corresponds to searching the locale structure for vertices (equivalently lines or curves) which are surrounded by distinctive patterns. The structure of the locale network makes this computation relatively straightforward and efficient. The location of the agent is expressed in terms of a locale and its coordinate system

agent-location = (locale pose)

so it is known what locale the agent is inside. This locale, together with its offspring (freespace and the objects contained in the locale) , define the scope of what can be seen by the agent: the agent can see the inside of the locale and the outside of the offspring. The search for landmarks consists of collecting the vertices (and/or lines) which make up these surfaces and selecting those which are surrounded by distinctive patterns. The surface patches used as landmarks are the vertices (lines) together with a some surrounding area .

More precisely, the procedure for selecting landmarks from the model is as follows:

1. From the locale corresponding to the agent's current location collect all the vertices associated with the regions on the inside surface of the locale and the outside surface of each offspring locale. This is accomplished by looking at the surface description for each locale:

locale --> surface --> faces --> regions -- > lines --> vertices

2. Delete vertices which are not expected to be visible to the agent from its current location. This is done by first clipping the projection to the image plane (ignoring occlusion) and then deleting occluded vertices from what remains.

3. Discard vertices which do not touch two regions which differ in reflectivity by some threshold. These reflectivities can be found by following pointers to the associated regions:

vertices --> lines --> regions --> reflectivity

4. Return the remaining vertices.

Landmarks correspond to the vertices resulting from this procedure. These are the landmarks used in action-level and plan-level perceptual servoing. In both types of servoing, knowledge of the reflectances of these landmarks is used to construct the appearance templates used to identify the landmarks in the image data.

5.3 Constructing Appearance Templates

Appearance templates are image arrays which specify how a landmark should appear in the image. For the vertex landmarks described in the previous section, these are n x n image arrays centered on the image of the vertex with pixel values determined by the geometry, reflectance values and the agent's location as represented in the model. Constructing these arrays is a localized rendering process.

Templates are constructed by ray tracing, considering only those surfaces and regions whose boundaries pass through the vertex. The pixel values in the template array are assigned to be proportional to the reflectance value of that region which the ray strikes first. A more precise statement of this procedure is:

1. Collect the regions which form the vertex. This is accomplished by following pointers in the locale network, beginning with the vertex frame:

vertex --> lines --> regions

2. For each pixel in the appearance template there is a ray which starts at the focal point of the camera lens and passes through the center of the pixel. Find the 3D intersection of this ray with each of the regions collected in (1).

If the ray does not intersect any of the regions either the patterns associated with the landmark are too detailed and may be subject to aliasing or the vertex is on an occluding edge of an object and certain pixels will be unpredictable. The use of the landmark will be error prone so it should not be used. In this case a null template is returned.

Otherwise assign to the pixel the value 255*R, where R is the reflectance of the region whose point of intersection with the ray is closest to the camera lens focal point.

3. The resulting array is the appearance template.

The resulting template is used match the landmarks with their projections in an image. This matching is done using correlation in a way which matches reflectance values, rather than intensities, to remove the ever present effects of uneven illumination.

5.4 Matching Templates to the Data

Correlation is a well understood mathematical tool which has been widely used in signal processing and in 2D image processing, but it has not been used as much in 3D scene processing because there are several severe problems which arise. It is easy to describe these problems if we think of an image as a function f(x,y) on the x-y plane. Correlation is a measure of how similar two such functions are. Images of 3D scenes are, however, strongly effected by several factors, all significant in the kinds of scenes our agent will encounter.

1. The shape of the image of an object varies as the viewpoint is changed, due to projection effects.

2. Changes in viewpoint alter what can be seen in the image of a scene, due to occlusion effects.

3. Changes in lighting modify the intensity of the image, either locally or globally.

4. Specularities vary, sometimes strongly, with lighting and viewpoint

All four problems affect the nature of the image function f(x,y) corresponding to a scene in such a way that its "shape" and "height" may vary considerably. This makes correlation useless for global scene matching and makes local scene matching difficult, at best. Knowledge of the agent's approximate location, together with knowledge of how these problems affect the image function makes it possible to cope with these problems. Problems 1 and 2 are managed by the way the appearance templates are constructed. The perspective distortions of 1. and the occlusion effects of 2. are kept to a minimum by the ray tracing procedure and the rejection test in step 2 of the construction method described above.

The third problem is managed by correlating reflectance values, rather than intensity values. For sufficiently small patches in the environment it is assumed that the light intensity is constant. This is not an unreasonable assumption for an environment which is not highly textured by shadows. The values of the pixels P(i,j) corresponding to such a patch can be expressed as:

$$P(i,j) = I*R(i,j)$$

where I is the (constant) light intensity over the surface patch and R(i,j) is the average reflectance over the surface which contributes to the pixel value P(i,j). Under these circumstances the effects of the illumination can be removed by dividing each pixel in the image by the sum of the pixel values over the patch. The new array RP(i,j)

$$RP(i,j) = \frac{P(i,j)}{\sum_{k,l=0}^{n} P(i,j)} = \frac{I*R(i,j)}{\sum_{k,l=0}^{n} I*R(i,j)}$$

$$= \frac{I*R(i,j)}{I*\sum_{k,l=0}^{n} R(i,j)} = \frac{R(i,j)}{\sum_{k,l=0}^{n} R(i,j)}$$

consists of values which are independent of the illumination incident on the surface and depends only on the reflectance properties of the surface itself.

Management of the fourth problem, specularities, is still a matter for future research. Currently it is assumed that the number of landmarks unrecognized due to specularities will be minimal. This assumption seems to be well justified in our experiments to date.

Matching itself is done using normalized correlation. Normalized image patches RP(i,j) are matched against normalized appearance templates RT(i,j) using what is known as normalized correlation:

$$NC(i,j) = \frac{2*\sum_{k,l=0}^{n} RP(k-i,l-j)*RT(k-i,j-l)}{\sum_{k,l=0}^{n} RP(k-i,l-j)^2 + \sum_{k,l=0}^{n} RT(k-i,j-l)^2}$$

The implemented computation differs somewhat from this for efficiency reasons, but only in the order in which the elements are computed. This correlation method has proven to be very reliable for locating landmark images in the indoor environments used for the experiments described below. Outdoor experiments are planned for the near future.

Once the selected landmarks have been located in the image we have matched the image data to the 3D landmark models. Next appropriate corrective actions are taken. These actions are different for action-level and plan-level servoing, so we describe them separately.

5.5 Action-Level Perceptual Servoing

Actions are carried out incrementally, using the location of landmark images to compute necessary corrections Each increment begins by selecting landmarks and matching their projections with data in the image. By measuring the discrepancy between where the features should be and where they are determined to be in the image, it is possible to compute the corrective action required to bring the positions into agreement (Figure 7). This perceptual servoing has the effect of locking the robot onto a trajectory which improves the accuracy of the actions over that which would be obtained without servoing.

The "move" action illustrates the action-level perceptual servoing principle well. Experiments have shown that during an open loop execution of an intended straight line forward motion the robot vehicle moves in a curved path. Sufficiently small incremental paths can be approximated by a straight line at some angle with respect to the intended line. Figure 8 depicts the geometry of the situation.

If we let

m = incremental distance moved
e = distance "off desired line" after an incremental move of distance m
q = shortfall in distance covered along intended line.
x = x coordinate of the landmark image (in camera coordinates)
f = focal length of camera lens
X = x-coordinate of landmark in robot coordinates (expected)
Y = y-coordinate of landmark in robot coordinates (expected)
h = heading error after the incremental move.
b = measured landmark bearing

Then

$$h + b = \tan^{-1}\left(\frac{Y-e}{X+q}\right)$$

so

$$h = \tan^{-1}\left(\frac{Y-e}{X+q}\right) + \tan^{-1}\left(\frac{-x}{f}\right)$$

Thus from the observed x-coordinate of the landmark image we can estimate the heading error, provided the values of e and q do not distort the value of

$$\left(\frac{Y-e}{X+q}\right).$$

This is typically the case for small incremental motions. In our configuration e is on the order of .02 ft and q is on the order of .05 ft. Hence for landmarks more distant than 3 ft this ratio can be well approximated by (X/Y)

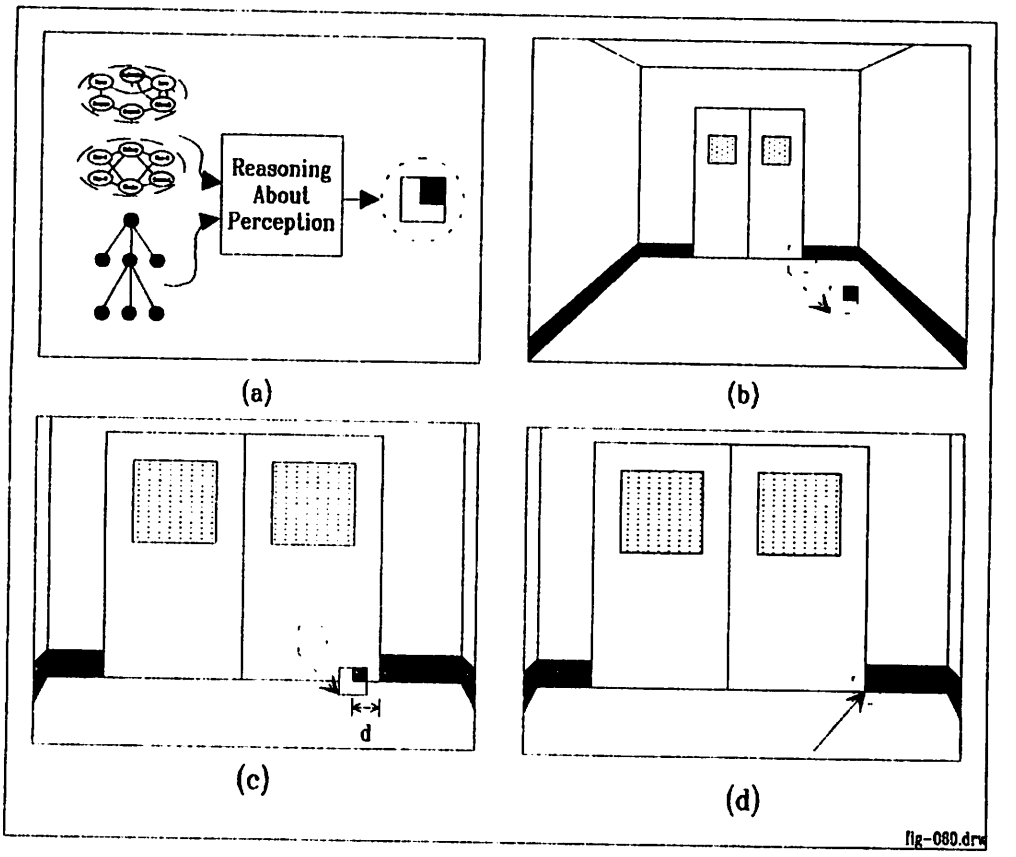

Figure 7. Before each action begins, landmarks are selected (a) using knowledge of the environment and the task. An incremental action is taken and the motion of the landmark is predicted (b). Any difference between this projection and the actual location of the landmark image (c) is used to determine what corrections to make.

Given an approximation to the heading error the corrective action is determined according to the geometry indicated in Figure 9. The quantities e and q are given by the equations:

$e = m*\sin(h)$

$q = e*\tan(h)$

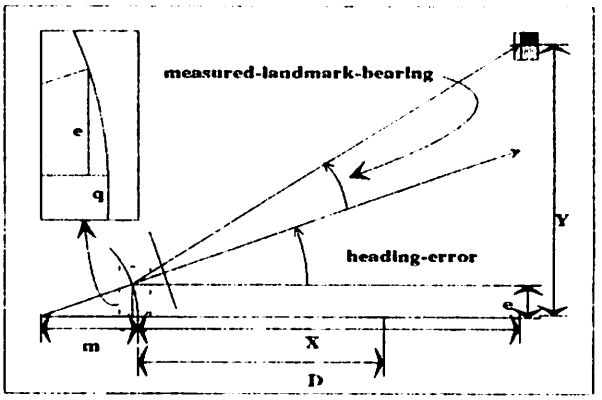

Figure 8 Experiments have verified that the motion of the robot can be approximated to be a straight line for very small distances. Under these assumptions the geometry of an incorrect motion allows us to compute the heading error from the position of a landmark in the image.

If the agent motion is corrected by steering to a point on the intended line a distance D away from the expected location we can correct as follows:

$$\text{correction} = -\left(h + \tan^{-1}\left(\frac{e}{D+q}\right)\right)$$

$$= -\left(h + \tan^{-1}\left(\frac{m*\sin(h)}{D+m*\sin(h)*\tan(h)}\right)\right)$$

The results of a simulation using this method are shown in Figure 10, both for a camera which is perfectly aligned with the robot motion, as was assumed in the above description, and for a misaligned camera. Aligning the camera to the robot motion can be a tricky operation, but it is only a modest effort to align within 0.005 radians, so the simulation predicts fairly accurate control.

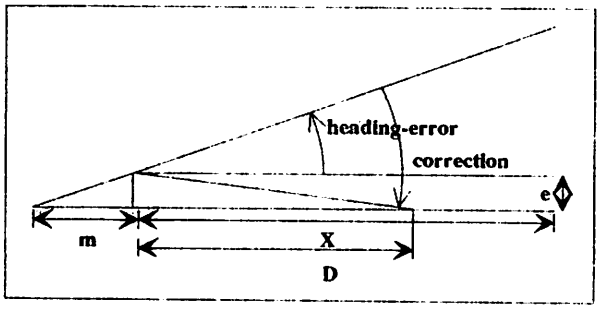

Figure 9 Once the heading error has been determined, the robot is turned toward a distant point on its original intended path.

To verify this prediction several experiments, both with and without action-level perceptual servoing, have been run. In the experiments Harvey was to roll along a straight line 40 feet long, marked on the floor of a Graduate Research Center hallway. The vehicle was stopped every two feet and its deviation from the marked line was measured. This experiment was run a number of times; the results shown in

the graph of Figure 11 represent the best unservoed result compared with a typical servoed result. Even after a rather painstaking setup procedure the unservoed vehicle wandered over two inches from the line within the first 20 feet. Other runs resulted in as much as a foot deviation in unservoed mode. Unservoed trials were stopped at around 20 feet because the vehicle was significantly off course and the total deviation was increasing. In contrast, in servoing mode the vehicle stayed within .3 inch of the line for the full 40 feet.

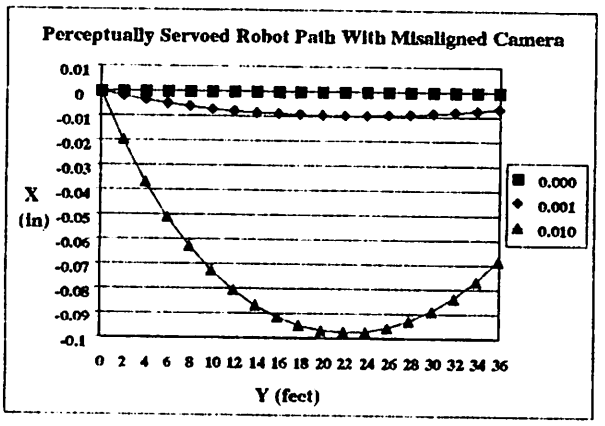

Figure 10. A simulation of the action-level perceptual servoing concept shows the effects of camera misalignment on motion control for 0.000, 0.001 and 0.010 radians of misalignment.

These experiments have only been performed indoors, but the results of these experiments are encouraging and support the use of perceptual servoing to control motion over reasonable navigation segments. It seems possible that each motion can be carried out accurately enough to support the assumptions made in the next section.

5.6 Plan-Level-Perceptual Servoing

Plan-level perceptual servoing uses the same landmark selection and matching procedures as action level perceptual servoing. Consequently, since the agent has been tracking the landmarks for steering purposes, it is likely that the matching effort during milestone recognition will be small, since the images of the landmarks at milestone recognition time will be known. The resulting matches and the 3D model information are used to determine the robot's location using a 3D pose refinement algorithm [KUM 89]. If the robot location as determined by the pose refinement agrees with what is expected then the milestone has been satisfied and there is no need to modify any subgoals.

If pose refinement and expectations differ by a small amount, then the first location in the next subgoal is adjusted to reflect this difference. Currently the next subgoal is changed from (ptrans location-1 location-2) to (ptrans pose-determined-location location-2). Plan-and-monitor automatically takes care of any detailed plan refinement if this change makes it necessary. Should the pose-determined location differ greatly from expectations, or if it should fail to determine a location, control is passed on to goal-level perceptual servoing.

To use position information in this way, how accurate must the pose returned by the pose refinement step be? Since the robot has been servoing on image features over the previous navigation leg, it is likely that it will be fairly close to its expected position, say within 6-12 inches. Within this area of uncertainty, pose refinement should return an accurate value, say within an inch or two. Outside of this area, it is likely that significant subgoal redevelopment will be necessary, so it is appropriate for plan level servoing to pass control up to goal-level servoing. Determining the robustness of combined action-level,

plan-level and goal-level servoing over a wide variety of environmental conditions is the subject of ongoing experiments.

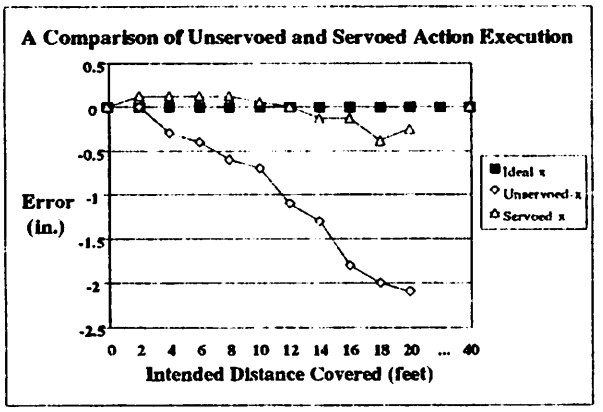

Figure 11. Experimental results have shown the ability to keep the robot within .3 inches of its intended course by using action-level perceptual servoing. Without servoing the robot wanders two inches from the path by the time it has traveled 18 feet.

Experimental results in support of using correlation matching and pose refinement for this purpose have been promising. Table 1 shows results from an experiment to determine the accuracy of this approach. Using a single image taken in a hallway, the expected location of the robot was varied to create the effect of being off target. The table shows the measured pose for different expected locations, corresponding to the actual location of the robot (y0) when the image was taken and three incorrect expected locations: to the left 3 inches (y0-3), to the left 6 inches (y0-6) and to the right 3 inches (y0+6) .

The experimental results discussed in sections 5.5 and 5.6 are quite encouraging. Each of the perceptual servoing algorithms seems to have significant potential for robustness when examined individually. How-ever, they effectively complement each other when combined into a system. The results of a multiple leg experiment in the hallway environment is described in [FEN 90]. In this particular experiment, the vehicle stopped within 1 inch of the final goal.

6.0 Conclusions

The preliminary indoor experimental results presented here support the use of perceptual servoing to increase the accuracy of primitive action execution over open loop execution and to maintain the position of the robot relative to its internal model. As weather permits, similar outdoor experiments will be performed. The results obtained from plan-level perceptual servoing using correlation matching compare favorably with a second technique using 2D line correspondences [FEN90] but is computationally less expensive. Both approaches use the 3D pose refinement mechanism developed by Kumar [KUM 89] to compute the pose and orientation of the robot with respect to the internal model from the landmark matches. The robustness of these techniques is currently being explored.

References

[BEV 89] 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.

[BEV 90] 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.

[BRO 86] 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.

[DIC 88a] E. Dickmans and V. Grafe, "Applications of Dynamic Monocular Machine Vision," Machine Vision and Applications, Vol. 1, pp. 241 and following. 1988.

	y0-6	y0-3	y0	y0+3
Expected				
x	40.00	40.00	40.00	40.00
y	3.50	3.75	4.00	4.25
z	0.00	0.00	0.00	0.00
theta-x	0.00E+00	0.00E+00	0.00E+00	0.00E+00
theta-y	0.00E+00	0.00E+00	0.00E+00	0.00E+00
theta-z	3.14E+00	3.14E+00	3.14E+00	3.14E+00
Measured				
x	39.93	39.93	40.11	39.85
y	4.01	4.01	4.08	4.17
z	-0.22	-0.22	0.20	0.00
theta-x	-4.69E-03	-4.69E-03	-9.00E-03	-4.95E-03
theta-y	-1.51E-02	-1.51E-02	-5.00E-03	-9.25E-03
theta-z	3.14E+00	3.14E+00	3.14E+00	3.14E+00

Table 1. Using correlation for landmark matching, together with 3D pose refinement experiments have shown the ability to determine the robot's location to within 1.5 inches. These results compare the expected (incorrect) location with what was measured. The ground truth location in all cases was the same x=40, y=4, z=0, theta-x=0, theta-y=0 and theta-z=3.1415.

[DIC 88b] E. Dickmans and V. Grafe, "Dynamic Monocular Machine Vision," Machine Vision and Applications, Vol. 1, pp. 223-240, 1988.

[FEN 88] C. Fennema, E. Riseman and A. Hanson, "Planning With Perceptual Milestones 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.

[FEN 89] 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. Also appeared in Proc. DARPA Image Understanding Workshop, pp219-231, May1989

[FEN 90] C. L. Fennema, A. Hanson, E. Riseman, J. Beveridge and R. Kumar, "Towards Autonomous Navigation", IEEE Systems, Man and Cybernetics, 1990, to appear.

[HER 88a] M. Herman and J. Albus, "Overview of the Multiple Autonomous Underwater Vehicles (MAUV) Project," Proc. IEEE International Conference on Robotics and Automation, Philadelphia, PA, pp. 618-620, April 1988.

[HER 88b] 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.

[KUM 89] 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 more detailed version may be found in Dept. of Computer and Information Sciences, University of Massachusetts (Amherst), TR89-120, December 1989.

[LOW 85] 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.

[SCH 76] R. Schank and R. Abelson, "Scripts, Plans, Goals and Understanding", Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1976.

[THO 86] 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.

[TOS 87 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.