CHAPTER VI

PERCEPTUAL STRATEGIES FOR MOBILE ROBOT NAVIGATION

In order for a mobile robot to be able to navigate intelligently in an only partially modeled world, environmental sensing is necessary. The previous chapter described the role of motor behaviors in driving a vehicle to satisfy its navigational goals. The importance of embedded perceptual strategies (or "identification procedures" [6]) for action-oriented perception as the means for producing these behaviors should now be clear.

The AuRA design (Fig. 2) ideally includes the VISIONS system being developed at the University of Massachusetts under the guidance of Hanson and Riseman [53,55] as the sensory gateway. Due to the extremely high computational requirements of sensory processing and the continually evolving but partially incomplete status of VISIONS, AuRA's initial implementations of necessity require reliance on individual vision algorithms drawn from within that framework rather than the entire VISIONS system itself. These modular algorithms have been modified to come closer to efficient real-time performance than would otherwise have been available. Additionally, they draw upon top-down knowledge and expectations as provided by LTM and/or previous images. Action-oriented perception is the key concept employed in their modification.

Certain common threads run throughout these algorithms. The clear separation of a "start-up" phase from the "update" phase can be seen in most of the visual strategies. Tuning of an algorithm's expectations on a frame-to-frame basis are made based on current environmental conditions, such as lighting, relative robot position, and the like. By providing for adaptability in the tracking of image features, whether they be obstacles, paths, or landmarks, the goal is robust feature recovery. Other work [137,134] in visual navigation uses similar techniques.

Although vision is a principal concern of AuRA, other sensor modalities are exploited where available. Ultrasonic data, available from the ring of 24 sensors surrounding HARV,

provides information regarding the distance to surfaces. Although the discriminatory capabilities of ultrasonic data is limited, it serves a useful purpose in obstacle avoidance and confirmation of visual interpretations.

Shaft encoders, measuring the distance traveled and the amount of rotation of the vehicle, provide limited sensor information. Chapter 7 describes the use of encoder data to manage spatial uncertainty growth. Distances traveled can be approximated using these sensors as long as the realities of their limitations are incorporated into the system and its representation of uncertainty. Shaft encoder data does not truly involve environmental sensing. It only records the number of rotations of the robot's motors, not the changes in the robot's position relative to the world. These limitations are discussed later in this chapter as well as in Chapter 7. An inertial navigation system would be highly preferred if available or its cost could be justified. Unfortunately, neither is true for our experimental environment.

The balance of this chapter is divided into the following sections. Section 1 discusses the potential relationship between VISIONS and AuRA. Section 2 describes the practical short-term role of specific modular vision algorithms used within AuRA. These include a fast line finder, a fast region segmenter, a depth-from-motion algorithm, and interest operators. Section 3 describes the ultrasonic algorithms in use in AuRA, including obstacle avoidance, door finding, panic sensing, and localization. The limitations and use of shaft encoder data appear in Section 4. Section 5 briefly describes other desirable sensors that may be valuable for potential integration into AuRA. Section 6 discusses some of the implementation details for AuRA's perception subsystem. The chapter then concludes with a summary and evaluation of the role of the different perception techniques currently employed in AuRA.

§1. VISIONS and AuRA

Scene interpretation has long been a primary research effort within the VISIONS group at the University of Massachusetts. Considerable literature exists describing the progress to date [102,140,54,53,55,41]. The remainder of this section will first briefly describe the operation of the schema system, followed by the role that the schema system can play in mobile robot navigation. It should be understood from the onset that schema-

based scene interpretation is currently a very time-consuming process. Work is underway, however, to provide parallel hardware (the UMASS Image Understanding Architecture [55,139]) to speed up this process by several orders of magnitude. Additionally, available *a priori* knowledge present in LTM can be used to guide schema instantiation and reduce the processing requirements dramatically.

§1.1 *The Schema System*

The VISIONS schema system accepts an image as input and produces a labeled interpretation of the observed environmental objects (Fig. 55) and, to the degree possible, a 3D representation of the environment. There are three levels of processing available utilizing both bottom-up and top-down processing (Fig. 11). Taking a bottom-up view first, the low-level processes operate on pixel level data producing an intermediate symbolic representation of line, region, and surface tokens. At the highest level, schema processes exist which interpret and collect the intermediate representations into labeled objects.

If no top-down guidance were available, it would be virtually impossible for the system to converge on an acceptable interpretation. Perceptual schemas (in the context of VISIONS) post hypotheses about what specific image events mean. Each highly rated hypothesis guides intermediate and low-level processes in an effort to find self-supporting evidence. This top-down guidance brings the intermediate and low-level processing requirements down to tolerable levels. If the hypothesis cannot find sufficient support or is contradicted by other data, it is deinstantiated. On the other hand, if sufficient support for a hypothesis is available, that particular portion of the image will be labeled as being associated with a particular environmental object and inference mechanisms can direct further semantic processing.

It is quite difficult to describe the operation of the schema system in a few paragraphs. It is hoped that the interested reader will refer to the more comprehensive descriptions cited above [esp. 55,41] for a better understanding of its operation.

§1.2 *Utilization of VISIONS Schemas in Mobile Robotics*

The principal test domains to date for VISIONS schema-based scene interpretation have been house scenes and road scenes (Fig. 55). These efforts have been predominantly concerned with full scene labelings with no expectations of specific instances of object

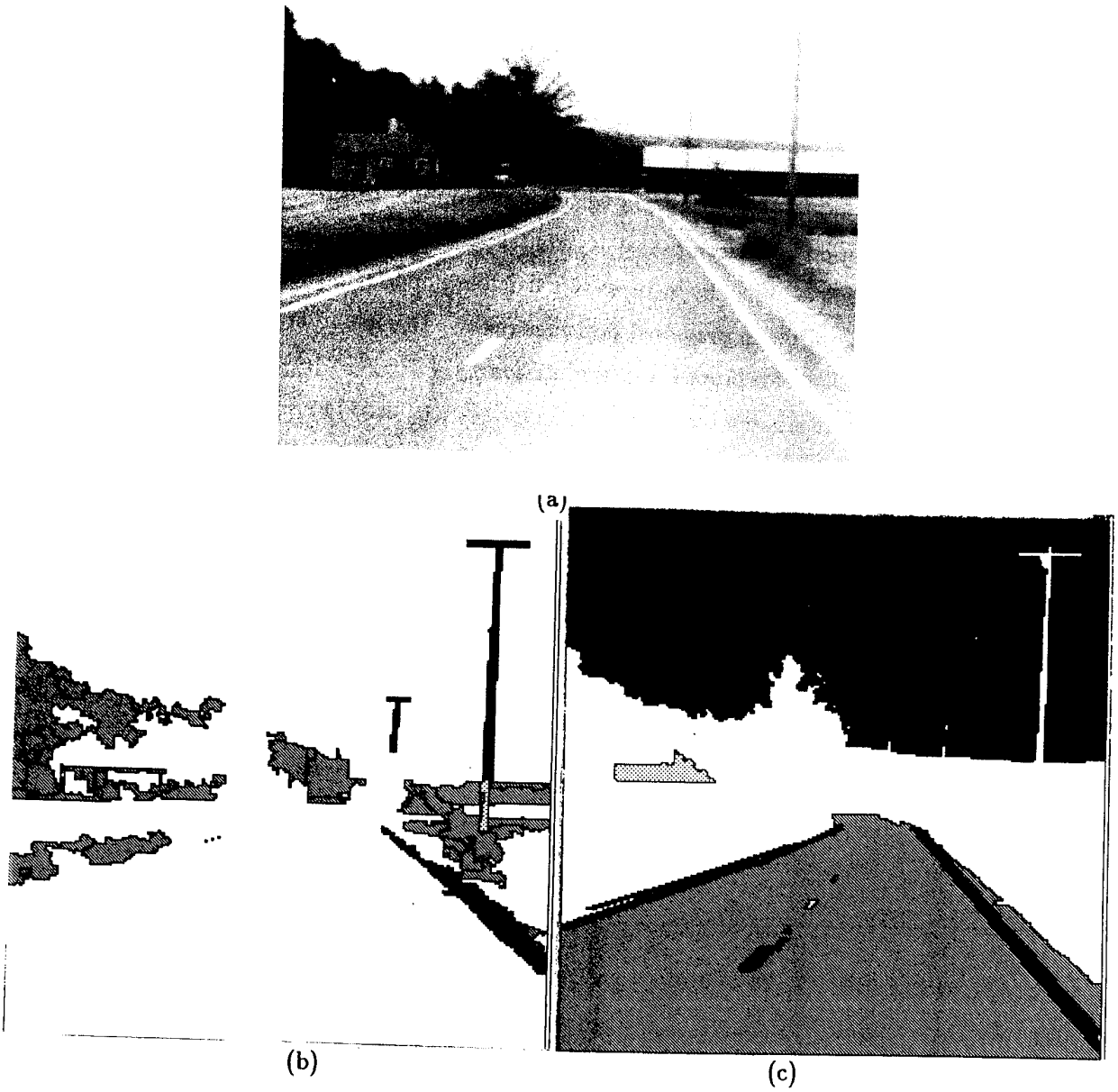


Figure 55: Schema-based scene interpretation

a) Original image.

b) Foliage, tree trunk, telephone pole, and gravel.

c) Sky, roof, road, and roadline.

classes in the particular image or environment in question other than it having such examples present; thus a general house or road scene is expected, but there is no world map of the domain.

If *a priori* knowledge of a specific environment is available, it can guide the posting of schema hypotheses, increasing the reliability of object recognition and reducing the amount of computing time required to achieve a satisfactory labeling. If the robot's position is approximately known within a global map, this information regarding the potential position of environmental objects (e.g. landmarks or roads) can be used to restrict the formation of object hypotheses to particular portions of the image. The occurrence of two known objects in predicted positions relative to each other can significantly increase the plausibility of a proposed interpretation.

Where can schema-based scene interpretation be used in mobile robotics? In the most grandiose sense, one can say for everything. If a completely and correctly labeled image is available, it can be used for navigation, obstacle avoidance, localization, goal recognition, etc. Indeed several of the other algorithms described below (line finding, region extraction, etc.) actually constitute some of the lower level processes used within the VISIONS system. Being realistic however, one must recognize that real-time responses are necessary for mobile robot navigation, indicating that schema-based scene interpretation is presently too slow to be effective. A more appropriate current use of the VISIONS schema system would be to provide for the top-down extraction of semantic objects of interest required for several of the other visual processes. If the initial image is analyzed by the scene interpretation mechanisms, it could yield the road edges that can be used to bootstrap the **stay-on-path** motor schema and **find-path** perceptual schema. Additionally it could provide the initial region statistics to seed the region extraction algorithm for path-following and landmark or goal recognition. Start-up information for the depth-from-motion algorithm could also be provided, in addition to potential corners that are of use for localization purposes by the interest operator. Finally, if the robot becomes sufficiently disoriented relative to its global map, the schema interpretation system could be invoked to enable the robot to regain its bearings relative to the modeled world.

§2. Modular Vision Algorithms

Although nothing in AuRA restricts sensor processing to be predominantly visual, much of the architecture is constructed to utilize this form of sensing. Action-oriented perception is the fundamental premise on which motor schema sensing and navigation is based. It is not necessary for the robot to fully understand the entire scene before navigation can be initiated (although this would certainly make things easier). Instead, by directing specific sensing strategies and the available computational resources to the motor needs of a particular task, only those portions of the scene which can contribute to the attainment of the pilot's goals are analyzed. Particular sensor algorithms are chosen to fit the demands of the specific path leg at hand.

No single perception algorithm is a panacea for navigation. The designer's goal instead is the development of a wealth of visual and other sensing algorithms which can provide the breadth that multi-domain navigation requires. A design goal of AuRA is to provide navigational capabilities in both indoor and outdoor environments, allowing for considerable environmental diversity in each of these cases.

Computationally efficient vision algorithms are used to provide navigational information for the robot and are initially implemented on a single processor. In later implementations specific processors will be dedicated for each algorithm to improve performance and eventually the load will be distributed over parallel hardware.

From an experimental point of view, this architecture affords the flexibility to try new perceptual strategies without forcing significant changes in the supporting system components. By embedding motor actions as behaviors and perceptual strategies as focus-of-attention mechanisms, both represented in a schema form, the addition, modification and deletion of these program units is manageable. The emphasis is on modularity. New world representations can be embedded in LTM through the use of the feature editor, providing for representational extensions that may be needed by new algorithms.

Typical of many of the algorithms is their ability to be decomposed into two phases: start-up (bootstrap) and update (feedforward). The start-up phase performs more slowly and has less, if any, *a priori* knowledge to work from. The start-up process produces initial region seed statistics, depth information, line orientation, etc., for establishing expectations which are used to advantage in subsequent frame analysis. The update stage uses

the information provided from the start-up phase to restrict the possible interpretation of image events and limit the search area for those events, thus reducing processing time significantly. The initial output of the start-up phase is updated after each processing run and is fed forward to provide a basis to guide analysis of the next image.

The remainder of this section will discuss some of the sensor algorithms that exist for use within AuRA. The strategies described below are not exhaustive, but rather represent the current initial elements being introduced by VISIONS researchers for use within AuRA's framework. It should be noted that a vision algorithm by itself is useless for navigational purposes. Considerable amounts of additional software must and has been created in order to produce intelligent motion of a robot vehicle using that algorithm. Each of the algorithms described below has been used for navigation experiments with HARV (Chapter 8).

§2.1 *Line Extraction*

Line extraction has the potential for multiple uses within AuRA. These include path edge extraction for use by **stay-on-path** schemas, landmark identification for **find-landmark** schemas, and as a texture measure for terrain identification. Of these, the first two are currently being developed for use in AuRA. The remainder of this section will first describe the fast line finding algorithm, and then its application to both path following and localization purposes.

Fast Line Finder (FLF)

A fast line finder based on Burns' algorithm [31] has been developed by Kahn, Kitchen and Riseman [64]. It is a two-pass algorithm which first groups the image data based upon coarse quantization buckets of gradient orientation into edge-support regions. This grouping process collects pixels of similar gradient orientation into separate regions via a connected components algorithm. The gradient magnitude does not affect the line extraction process. A line is then fitted to the resultant edge support region. FLF differs from the original Burns' approach by permitting the specification of the gradient orientation buckets and the extraction of the representative line for each edge support region. Many of the elementary computations can be further speeded up through the use of a conventional pipeline processor which supports a look-up table and convolution

processing. An outline of the algorithm appears in Appendix C.

Fragmentation of a potentially long image line often occurs if no *a priori* knowledge is available regarding the approximate orientation of the line in the image. The likelihood of extracting a particular long line increases by tuning the bucket's orientation to be centered on the anticipated orientation of a road edge or other line model in the image through the use of available knowledge extracted from LTM or previous images.

The key concept is *action-oriented perception*, performing only that computation which is necessary for the specific task at hand. Features available within the FLF algorithm to support this concept include the ability to scope the image (i.e. perform line extraction on a subwindow of the image). If the robot has approximate knowledge of the world position (and hence image position) of the line feature being sought (derived from LTM, the spatial uncertainty map, and/or previous images), substantial processing reductions are attained by ignoring those portions of the image where the feature is unlikely to occur. In addition to orientation, the FLF can be adjusted to filter lines based on gradient magnitude, dispersion, size of the region, and length. By adjusting these filters in advance, based on the features desired (e.g. short lines for texture, or long lines for roads), unnecessary processing is minimized. A secondary filtering procedure is also available for removing lines after the fast line finder has been run, making it possible to collect different sets of lines with different characteristics from only a single run of the more time-consuming FLF. This is possible because when the lines are produced, statistics regarding each line are collected and stored with the endpoint data for later reference.

Figure 56a is an image of a sidewalk scene. Figure 56b shows the results of the FLF using the full default set of coarsely quantized orientation buckets for the entire image. Figure 56c shows the results with the orientation buckets tuned and the subimage scoped to the anticipated road edge based upon the internal model of the vehicle position and orientation, while Figure 56d shows the results with the buckets tuned to horizontal and vertical edges, filtering to retain longer lines and with the image scoped above the horizon.

Path Following

A significant contribution of this dissertation involves the application of the FLF to path following. Using line finding to extract path boundaries requires the grouping of

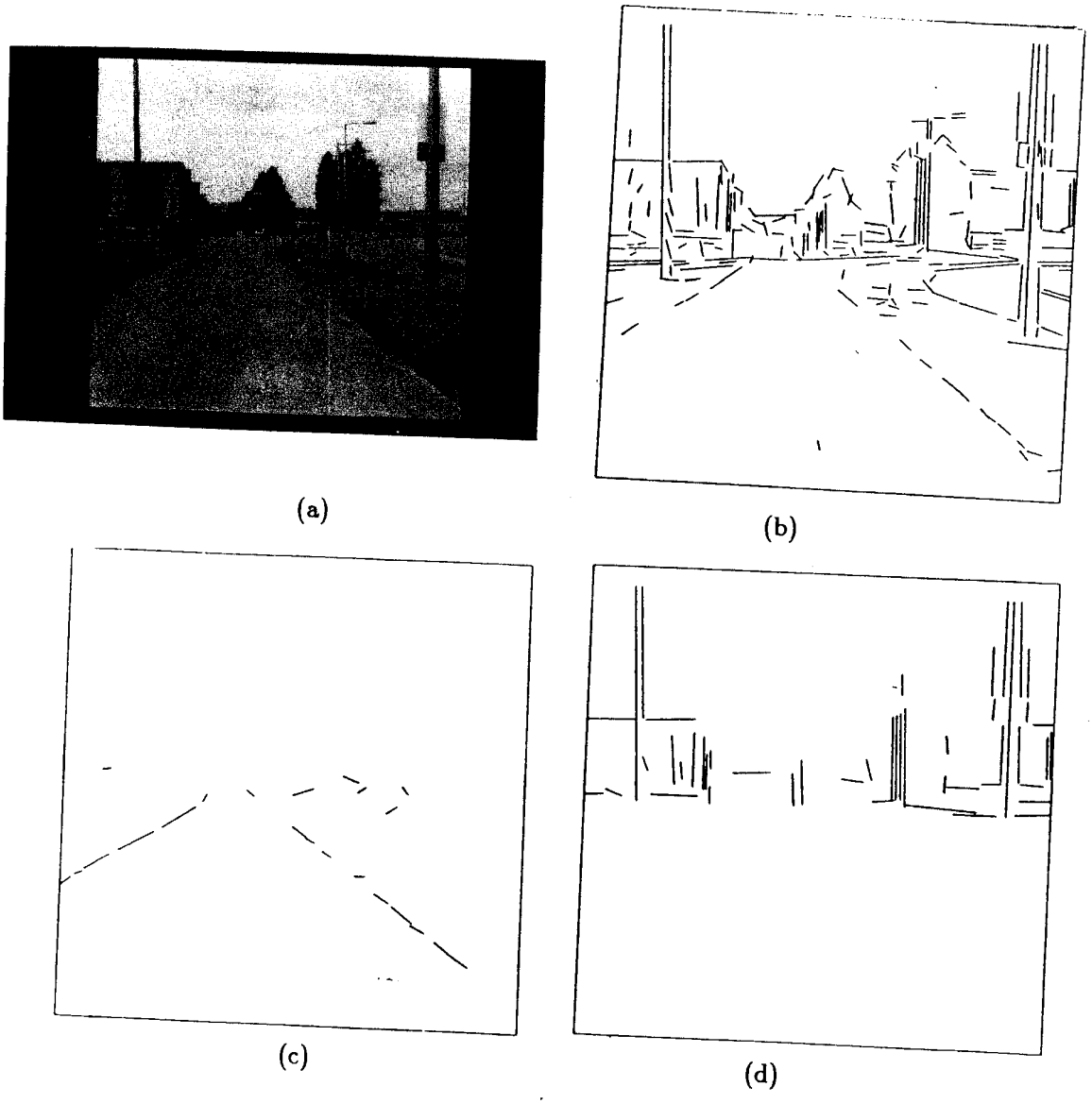


Figure 56: Fast line finding

a) Original sidewalk image.

b) Default bucket orientation.

c) Buckets tuned to road edges.

d) Buckets tuned to long vertical and horizontal lines above horizon.

resultant FLF line fragments into a single line representing each path edge. No effort is made to condition or modify existing paths to make this process easier (e.g. by adding stripes, cleaning, etc.). The grouping strategy used must be able to deal with fragmentation and edge discontinuities, such as path intersections, leaves, etc.

If the uncertainty of the vehicle is within reasonable limits, predictions of the position and orientation of the road lines in the image plane can be made. As described above, there are two distinct components of road-following (see also [137]): the bootstrap or start-up phase, where the road edge is determined in the image for the first time; and the feedforward or update phase - where a previous image is used to guide the processing for the next image. Line finding is not necessarily the best strategy for initially finding the road's position. Nonetheless, it can be reasonably effective if the road appears on a global map of the terrain and there is approximate information about the vehicle's position and orientation. These are both present within AuRA: in LTM and the spatial uncertainty map, respectively.

The feedforward phase assumes that the approximate position of the road was known in the last image. This information, when coupled with the commanded translation and rotation the robot has undertaken since the last image acquisition, can be used to predict approximately where and at what orientation the road edges will occur in the newly acquired image. As anyone who has worked with mobile robots knows, the motion that a robot actually takes may differ quite significantly from that which it was commanded to perform. Consequently, there must be a considerable margin for error in these predictions if the algorithm is expected to be robust. Additionally, there must be some measure of the confidence in the line produced representing the road edge.

Path edge grouping proceeds as follows: The buckets are tuned based on the anticipated position of the road edge in feedforward mode; in bootstrap mode either knowledge from LTM or the default buckets would be used. The fast line finder is then run, producing line fragments in the approximate orientation of the path edge (Fig. 57a). These fragments are then filtered based on their distance from the anticipated image line and the expected orientation of either the right or left edge. Again the amount of tolerance allowed is controllable. This yields two sets of line fragments (one for each path edge - Fig. 57b-c). All the fragments above the vanishing point of the road, (obtained from feedforward information), are discarded. The center of mass of the midpoints of

the remaining line fragments is computed, each midpoint weighted by the length of the fragments themselves. The average orientation is computed in a similar manner. The resulting point on the line and computed line orientation determine the line equation for each road edge. The left and right edges are then used to compute the road centerline (Fig. 57d). The centerline is the basis for determining the rotational deviation of the vehicle relative to the road's vanishing point as well as the translational deviation from the road centerline. These newly computed path edges are then used as the models for the next feedforward step.

The total length of the line fragments used in producing the path edges serves as a measure of uncertainty. If this value drops below a specified threshold, special processing is undertaken. This includes increasing the error tolerances and margins in the FLF to see if a more confident line can be extracted from the same image, or if that fails, to digitize another image in the event that a passing obstacle blocked one or both path edges. If both of these strategies fail, the robot will reposition itself slightly and try another image. If this yet fails, alternate bootstrapping methods must be brought to bear.

Although the FLF algorithm was written by other members of the VISIONS group, considerable work was required to produce a useful tool for mobile robot navigation. The feedforward mechanisms, line fragment grouping, centerline extraction, image sequence acquisition, and vehicle servoing routines all had to be produced before the algorithm was suitable for navigational purposes. This is typical of all of the vision algorithms described in this chapter.

The robot is able to successfully navigate both an outdoor sidewalk and an indoor hall using the FLF. Approximately 10 CPU seconds (VAX-750) are required for each step to provide the robot information for traveling 5.0 feet ahead. This is approximately two orders of magnitude faster than the original Burns' algorithm [31]. The 512 by 512 image digitized on a Gould IP8500 is averaged to 256 by 256 before line extraction. This time can be reduced by using pipelined hardware available on the digitizer. The vehicle servos on the computed centerline position, correcting both orientation and translational drift as it proceeds. See Chapter 8 for details of these and other line-finding navigation experiments.

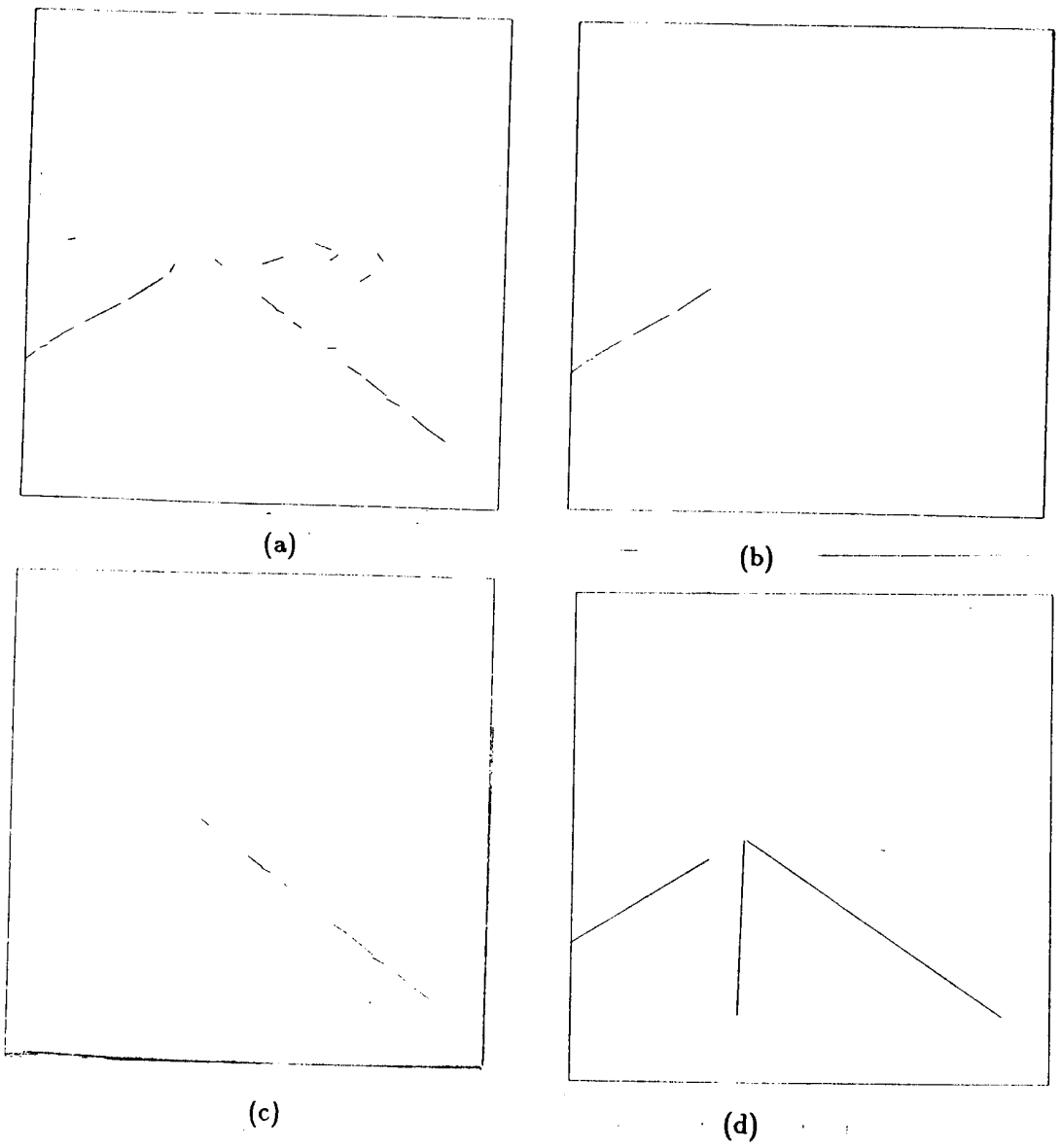


Figure 57: Path following with FLF
a) Output of FLF when run on image 57a.
(lines below horizon with orientation-tuned buckets).
b) Fragments left after filtering and windowing for left path edge.
c) Fragments left after filtering and windowing for right path edge.
d) Resultant path edges and computed road centerline.

Landmark Identification through Line Finding

Vehicle localization is addressed by the line finding algorithm using data stored in LTM. Localization is simply orienting the vehicle relative to its global map; in other words getting its bearings. It is not proposed that lines are the only mechanism for localization, but they should serve in conjunction with other relevant algorithms. In the role of a confirmation mechanism, or for tracking from frame-to-frame a previously identified landmark feature, FLF localization is well suited. Extracting the edges of a path as described above also provides information for localizing the vehicle, assuming the path is represented in the world map.

Long, strong vertical lines and corners derived from such lines are probably the most appropriate general category of lines suitable for this application. Edges of buildings, telephone poles, lampposts or doorframes can be tracked using the line finder. Figure 56d shows the result of running the FLF on image Figure 56a with the buckets and filters tuned for long horizontal and vertical lines of high gradient magnitude. This orientation can be used to identify the roofs of buildings against the sky or road intersections directly in front of the vehicle. By windowing the image for a certain landmark based on the position of the vehicle as indicated by the spatial uncertainty map and *a priori* knowledge of the global coordinates and dimensions of the object feature in question (from an environmental map in LTM containing object attributes and locations), it becomes possible to isolate features such as the corner of a building by combining the evidence from both horizontal and vertical lines. This then is used to constrain the positional uncertainty of the vehicle by backprojecting the 2D data to 3D world coordinates when combined with the knowledge of the height of the feature (see Chapter 7).

§2.2 *Fast Region Segmenter (FRS)*

FRS is a region extraction algorithm operating in a manner akin to the fast line finder, but based upon similarity of color and intensity features. It functions by first defining a look-up table that is used for classifying an input image. This algorithm has been motivated by histogram-based segmentation algorithms [67], but achieves great simplification via constraints from stored object knowledge in LTM or the result of processing previous frames, e.g. the look-up table ranges for specific objects may be defined on the basis of previous frames or from object data in LTM. The input image used can be an

intensity image, a gradient image, a color image component, etc. This input image is scoped (windowed) as in the case with the FLF. The look-up table maps ranges of pixel values to specific region labels. Available knowledge is used to define expected ranges of spectral attributes of interesting objects (Figs. 11,58,59). The resulting classified image is then subjected to a region extraction algorithm which groups the classified pixels into regions. Statistical data is then collected regarding each region. See Appendix C for an outline of the FRS algorithm.

The speed of this algorithm arises from the use of the look-up table to provide a quick mapping to the image. The connected components routine is then run on a restricted portion of the image selected through the use of top-down map constraints (see Chapter 7).

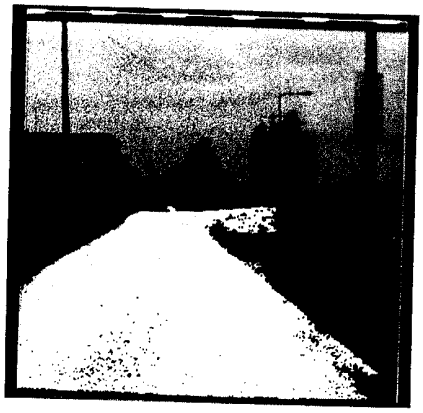
This segmentation is used for path extraction as in [124,128]. Preliminary experimentation using intensity images can be seen in Figure 58. Figure 58a shows the original image and Figure 58b the region extracted representing the sidewalk. The statistics collected for the sidewalk region are then used for providing the expectations (feedforward) for the next image in the sequence [as in 124]. Color was not used for this segmentation and the algorithm would be much more powerful with RGB input. Chapter 8 presents experimental results using FRS path following with HARV.

Landmark extraction is handled similarly. The centroid of the landmark can be used for localization purposes, in contrast to the edge detection methods used by the FLF or the corner detection approach used with the Moravec operator described below. A bright yellow road sign (very dark in the blue sensory band) is segmented for localization purposes in Figure 59.

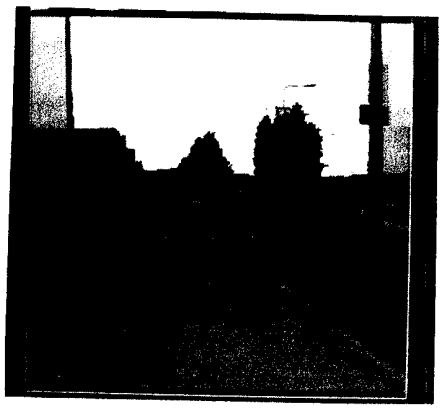
§2.3 *Depth from Motion*

Passive navigation by the determination of the position of environmental points via vision is an important sensor strategy for AuRA. The motion research group within VISIONS has long explored the extraction of depth from motion [142,73,2]. A more recent algorithm, developed by Bharwani, Riseman and Hanson [22], uses a sequence of frames under known translational motion of the sensor to incrementally refine positional estimates of objects over time. It can be used in mobile robotics for obstacle avoidance, position localization, and as evidence in object identification.

(a)



(b)



(c)

Figure 58: Sidewalk extraction via region segmentation

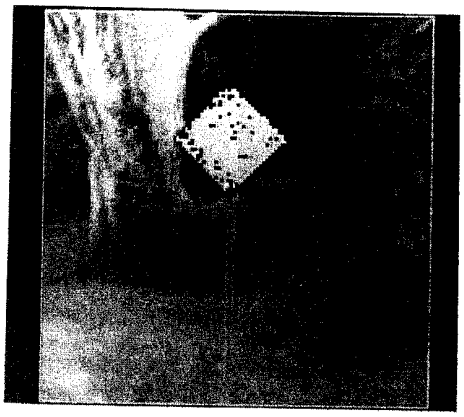
a) Sidewalk image.

b) Resultant extracted region representing sidewalk.

c) Resultant extracted region representing central portion of sky.

(a)

(b)



(c)

Figure 59: Landmark identification via region segmentation

a) Original sign image (combined RGB into intensity).

b) Blue plane of (a) chosen via LTM for analysis due to the spectral data of anticipated landmark.

c) Extracted region representing sign.

Algorithm

A brief sketch of the multiple frame depth-from-motion algorithm developed by Bharwani, Riseman, and Hanson follows. The reader is referred to [21,22] for the details of this approach. The algorithm allows refinement of depth over time up to some detectable limit, while maintaining a constant computational rate. This is very important for real-time processing.

The problem of recovering depth from motion in a sequence of images again involves the decomposition of the problem into two components: start-up and updating. This algorithm makes the assumption that the camera is undergoing pure translational motion and the position of the focus-of-expansion (FOE) is known within some reasonable estimated degree of accuracy.[1] This implies that the image displacement paths for a static environmental feature are constrained to move in a straight-line emanating radially from the FOE. An *interest* operator is used to extract *distinctive* points in the image, i.e. those of high curvature and contrast, that are likely to avoid false correlation matches in future frames.[2] It is assumed that the obstacles or landmarks will exhibit some such points on their boundaries. This is probable if the backdrop is bland (e.g. the road itself) or by deliberately retrieving only "interesting" landmarks from LTM. However, it should be expected that interest points will be extracted from both relevant and non-relevant image events.

The correspondence problem is the principal difficulty; how can one be sure that the feature in one frame corresponds to the same feature in the next frame after the robot has undergone translation? The start-up phase involves finding initial correct feature correspondences between the first two frames, while the update phase involves the use of the start-up analysis and the consequent approximate depth values to restrict the search area for corresponding matches at a higher match resolution in subsequent frames, thus reducing (or in general bounding) computation and providing refinements of the original depth estimates. Work by Snyder [119], addressing the limits of uncertainty in this type of motion, is fundamental to efficient use of previous correct correspondences in constraining

[1]These assumptions are not necessarily safe when using real world images. Frame registration and FOE recovery are problems that need to be solved. A discussion appears at the end of this section.

[2]See [2] for a method in dealing with the correspondence problem using a confidence measure. These ideas are implicit in the Bharwani algorithm.

the match in future frames.

Assuming the robot is traveling at a known velocity, the pixel displacements found between the first two images of a sequence (start-up) are used to further reduce the search for feature matching in successive frames, once the images have been registered so that non-translational motion of the camera has been subtracted out. Known sensor motion leads to a constraint on the match path, and approximate depth (from start-up) constrains the portion of the path to be matched. Progressive refinements can be made in the estimation of feature displacement and hence distance to a relevant feature.

Different strategies such as histogramming the collection of points on the basis of depth, determining orientation of surfaces based upon the depth of several points on associated regions, or identifying landmarks by correlating distance from the viewer with the objects in the environmental map in LTM, are all possible methods for extracting objects from the environment. The current approach for obstacle extraction is described in the subsection dealing with depth-from-motion system issues below. This data can then be used to provide information to the motor schema manager for effecting evasive action in the case of obstacles or for use in localization in the case of landmark location.

Applications

A primary goal of the depth-from-motion algorithm is to provide information about the distance of an object lying in the path of the robot. In obstacle avoidance applications, computational requirements are made tractable by restricting the processing to interest points (i.e. trackable image points of high contrast and curvature) and only to those that are lying within the current path of the robot.

Figure 60 and Table 1 illustrates some results using the depth-from-motion algorithm for obstacle avoidance. Chapter 8 discusses the experimental results obtained with this method using HARV. The biggest problems encountered in the use of this algorithm in mobile robotics include first, accurate recovery of the FOE, which can be minimized through accurate calibration of the camera relative to the robot, and second, ensuring registration of the images. Stabilizing the camera with a gyroscopic platform affords a hardware solution to the registration problem. A software solution [106] can be partially achieved by registering the images via correlation matching using points near and above the horizon, i.e. distant features (hence relatively unmoving with respect to the modest

amount of camera translation, thus any image translation or rotation observed can be assumed to be a consequence of improper image registration) that can be registered from frame to frame. Large rotations pose a particular problem and require many distant interest points and significant computation. The depth-from-motion algorithm is quite sensitive to misregistration due to rotation in the image plane, so every effort is made to minimize or eliminate any roll movements of the camera relative to the scene. Additionally, the FOE can only be extracted up to one degree of accuracy causing errors up to ± 5 pixels in a 256 by 256 image. This causes error in the returned value for depth, although the point tracking itself is generally unaffected. Extraction of depth in this manner is a difficult although promising problem.

The motion algorithm can be used for landmark identification as well. This is actually a simpler task than obstacle avoidance in many respects due to the availability of LTM knowledge to guide processing in a top-down manner. Knowledge of the approximate distance of a landmark to the vehicle in a restricted portion of the overall image substantially reduces the computation required. When approximate ranges for the distance to an obstacle are known, the algorithm will perform more robustly than when underconstrained. Portions of the image can be searched that are outside the obstacle avoidance regions. As these are usually further from the FOE than points in the robot's direction of motion, greater pixel displacements will occur and hence better results in the depth analysis.

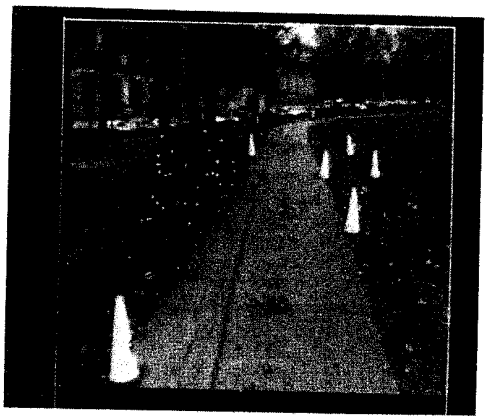
Depth-from-motion system issues

The depth-from-motion algorithm requires considerable support from other vision algorithms in order for it to be used for obstacle avoidance. The overall flow of control of the different components is shown in Figure 61. These components consist of a standalone Moravec interest operator, extract-focus-of-expansion (FOE), depth-from-motion, and obstacle extraction algorithms. It is also desirable to have a registration algorithm if the images are not acquired from a stabilized platform. Each of these components will be described in turn.

The Moravec operator is used to find the initial "interesting" points in the first incoming image. These points are used for establishing correspondences with other points in future images via intensity correlation of subwindows after the vehicle has undergone

(a)

(b)



(c)

Figure 60: Depth from motion

a-c) Three image sequence (distance traveled is approximately 4 feet between frames). Figures b) and c) show the corresponding tracked points. The results for the interest points are presented in Table 1.

(Image sequence taken from CMU NAVLAB).

Table 1: Depth from motion results

object	feature	nominal depth and uncertainty (± pix)			true depth (feet)		
		1-3	3-5	5-7	$frame_1$	$frame_3$	$frame_5$
cone1	1	76.38 ± 25.54	77.29 ± 16.96	72.48 ± 7.05	76.00	72.00	68.00
	5	90.16 ± 30.18	66.14 ± 10.65	62.07 ± 4.46	76.00	72.00	68.00
	6	92.63 ± 30.07	70.43 ± 11.66	61.33 ± 4.25	76.00	72.00	68.00
cone2	2	66.45 ± 16.57	90.93 ± 20.66	69.30 ± 5.61	76.00	72.00	68.00
	7	84.04 ± 22.51	82.15 ± 13.60	67.06 ± 4.46	76.00	72.00	68.00
	8	87.30 ± 25.19	77.68 ± 13.03	67.94 ± 4.74	76.00	72.00	68.00
cone3	3	53.12 ± 9.39	55.54 ± 6.47	47.44 ± 2.23	56.00	52.00	48.00
	11	59.84 ± 10.04	53.62 ± 5.17	47.73 ± 1.91	56.00	52.00	48.00
	12	54.80 ± 8.22	50.68 ± 4.48	46.05 ± 2.08	56.00	52.00	48.00
cone4	4	80.81 ± 26.83	60.07 ± 9.74	48.07 ± 2.93	56.00	52.00	48.00
	13	58.23 ± 10.88	58.56 ± 7.06	46.16 ± 2.06	56.00	52.00	48.00
	14	57.06 ± 10.38	53.97 ± 5.93	45.60 ± 2.73	56.00	52.00	48.00
can	9	48.21 ± 6.39	44.97 ± 3.52	38.54 ± 1.19	46.00	42.00	38.00
	10	45.75 ± 6.32	46.74 ± 4.18	39.92 ± 1.41	46.00	42.00	38.00
	16	44.78 ± 4.38	44.38 ± 2.83	37.65 ± 0.94	46.00	42.00	38.00
	17	46.84 ± 5.24	45.39 ± 3.45	39.01 ± 1.09	46.00	42.00	38.00
cone5	15	38.30 ± 4.25	35.86 ± 2.34	27.36 ± 0.79	36.00	32.00	28.00
	18	40.01 ± 4.38	33.22 ± 1.65	28.50 ± 0.56	36.00	32.00	28.00
	19	35.94 ± 3.26	32.79 ± 1.50	28.43 ± 0.53	36.00	32.00	28.00
cone6	20	20.34 ± 0.63	16.17 ± 0.22	14.17 ± 0.15	21.00	17.00	13.00
	21	20.53 ± 0.54	16.91 ± 0.21	* ± *	21.00	17.00	13.00
	22	19.95 ± 0.48	17.88 ± 0.22	* ± *	21.00	17.00	13.00

These tables contain the results for the images from Figure 60. (from [22])

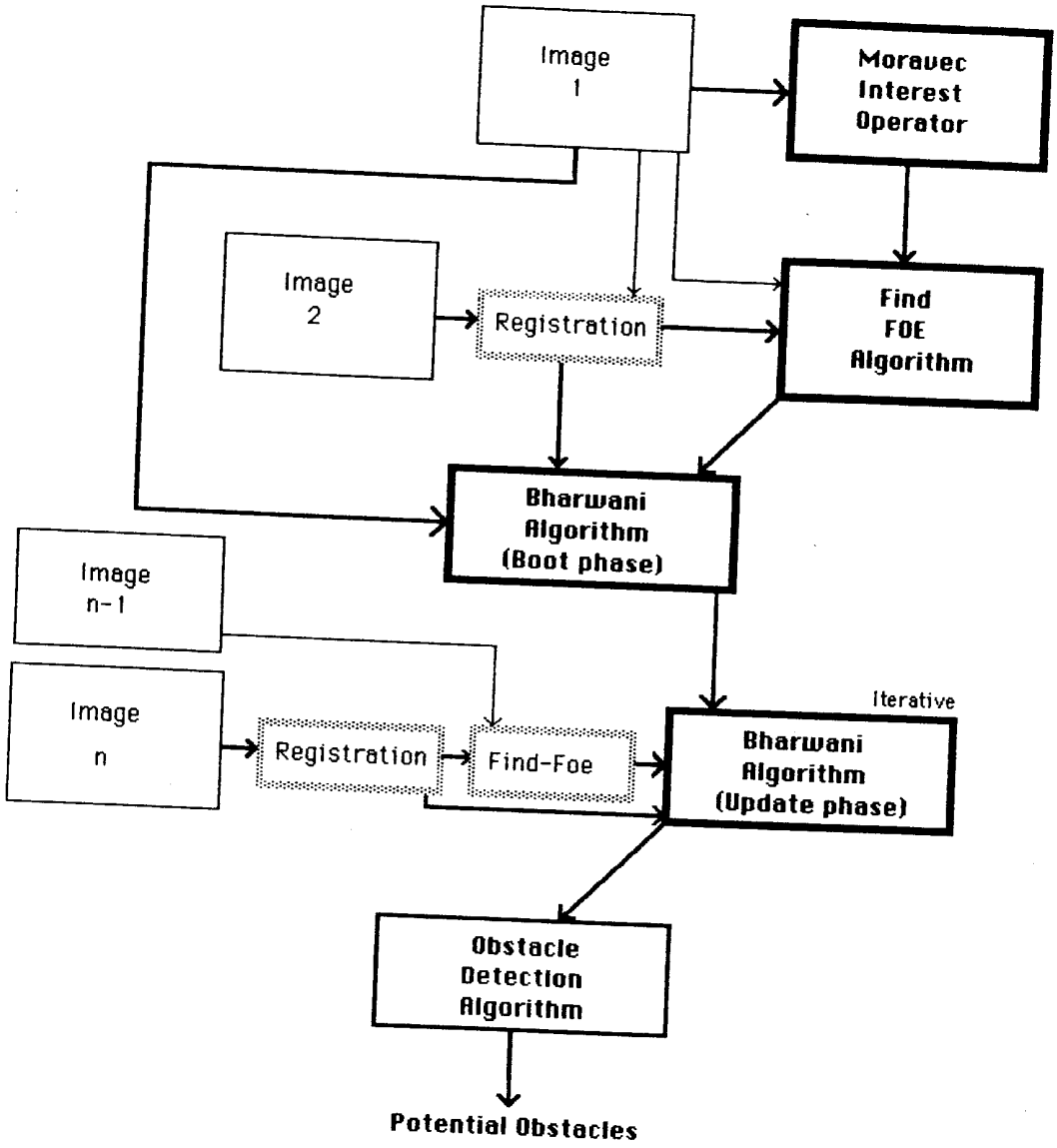


Figure 61: Depth from motion system

The fuzzy boxes (registration and second find-foe) are desirable components for the overall system, but are not automated in this implementation. Image registration is by hand and the FOE is assumed constant for the duration of the sequence.

translation. At this stage, the portion of the image in question is restricted to only those areas of interest for obstacle detection. This window is maintained throughout the rest of the overall system processing. The number of points extracted is user controllable as well; the fewer the points, the less processing time in making correspondences, but of course the less robust the results might be.

The second image now arrives, the robot having moved an approximately known distance. At this point, a registration algorithm, if available, is used to remove any translations or rotations of the camera that are not along the direction of translation. These can arise from bumps or rough spots in the road, eccentricities in the tires, etc. Registered images should be used throughout the rest of the processing.

The two available images are now presented to the find-FOE algorithm. This program [105] determines the focus of expansion for two successive images. If pure translational motion can be assumed for the sub-sequence of future frames, then the find-FOE algorithm need only be run once, as the same FOE will be present in all subsequent images. As this is not necessarily the case, the find-FOE algorithm may have to be run between all pairs of incoming frames, to allow for the movement in the position of the FOE.

The depth recovery algorithm is then run. Iterative refinement of depth occurs as each new image is acquired because the previous value of depth more tightly scopes the displacement in future frames which are matched at a higher correlation resolution. After a prescribed number of images and/or translational motion, the depth-from-motion algorithm transfers its tracked points and associated depths to the detect-obstacle module.

The detect-obstacle component, coded by the author, makes the assumption that the tracked points returned contain environmental points both on the obstacles and on the ground plane. An assumption is made that the area in front of the vehicle can be reasonably approximated by a ground-plane and that the vehicle itself is on that plane. A least squares line-fit is made to a plot of the row of the image versus the inverse of the depth (Fig. 62). This takes advantage of the perspective transform and its relationship to $1/Z$. If all the tracked points were located on the ground plane and the depths returned were accurate, all of the points would fall on this line. A full least squares plane-fit can be made to the points in three-dimensional space, but in the experimental runs used thus far it appears unnecessary. The points above the line are the potential obstacles, those points furthest above the line being the most likely obstacles. Essentially if a point in

an image row is closer than a point in that same image row which is on the computed ground plane (based on the least squares fit), it is marked as a potential obstacle. In other words, points on potential obstacles which are off the ground return closer depths than do other points on the same image row which are on the ground plane. The taller the obstacle, the greater the difference in depth between the obstacle's point and a point on the ground plane on the same row. Thresholding, based on the distance of the point from the least-squares fit line, is then performed on the candidate points (typically the 30-50% of the points with the greatest difference in the fit to the line that are above that line), returning the obstacles (Fig. 63). Occasionally false positives arise, but usually all the close to mid-range obstacles are detected. An alternative approach would be to determine several points on an extracted region and use those points to compute surface orientation; when such orientations are found to be vertical they represent potential obstacles.

The returned obstacle data can then be associated with **avoid-static-obstacle** schemas and used within the confines of the motor schema manager for navigational obstacle avoidance. Unfortunately, the algorithm is too slow on our current hardware to be used for real-time navigation. See Chapter 8 for details of the off-line experiments using this system.

§2.4 *Interest Operators*

Interest operators are used in computer vision to pick out pixels associated with regions of high curvature and contrast. The Moravec operator [93] and the Kitchen-Rosenfeld gray-level corner detection interest operator [65] are two well-known examples. The depth-from-motion algorithm, described in Section 2.3, uses an interest operator (currently Moravec's), to determine the points on which to run the correspondence algorithms for registration from frame to frame, and the points to initiate correlation tracking in future frames.

Interest operators are quite primitive as a stand-alone method for obtaining information for navigation. Their primary advantage is speed. By combining knowledge available from long-term memory with image data, it becomes possible to use interest operators to *confirm* the position of landmark corners. A clear-cut example would be the position of a building corner against the sky (Fig. 64). When combined with knowledge from the robot's spatial uncertainty map and object size and location from LTM, this method can

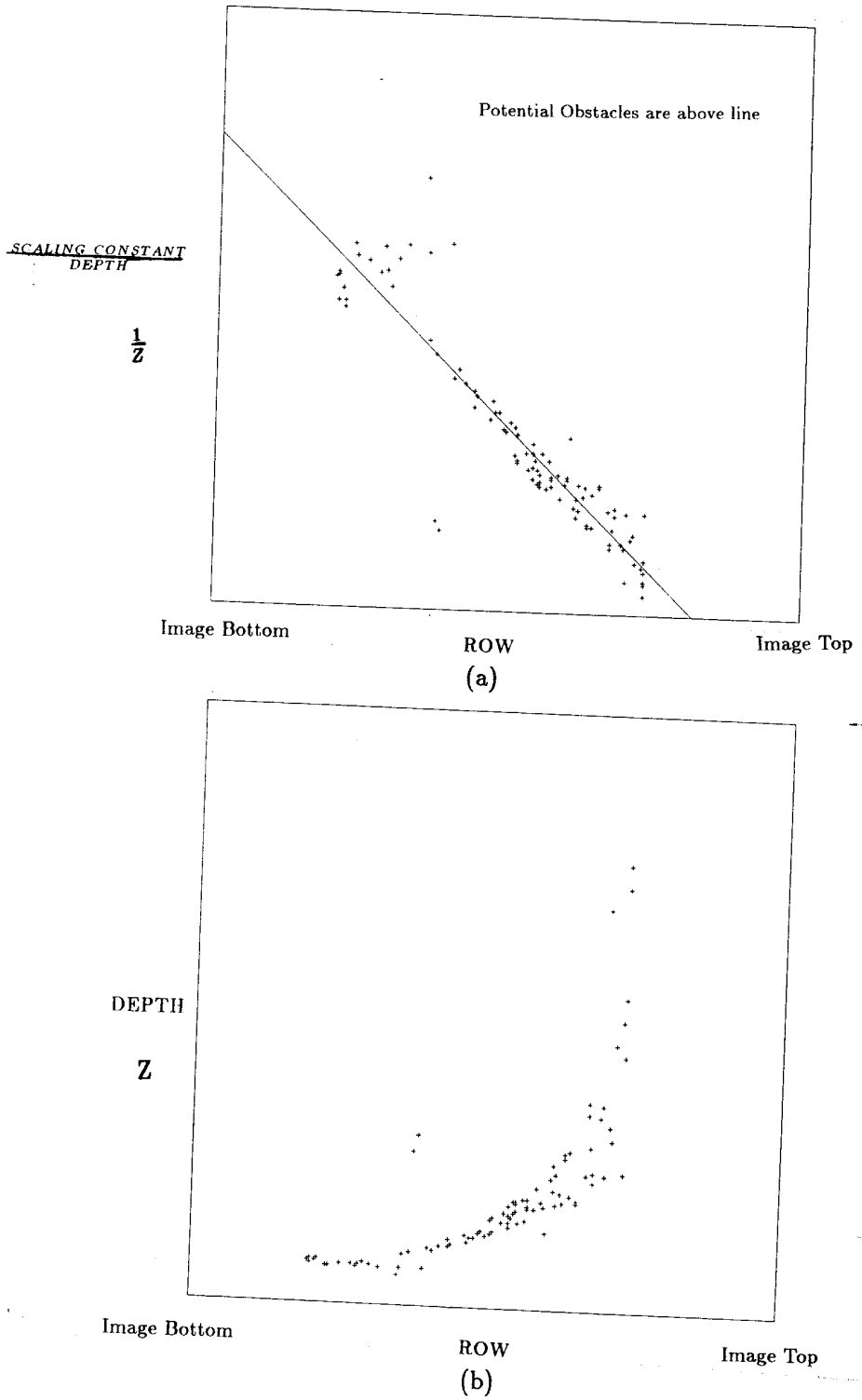


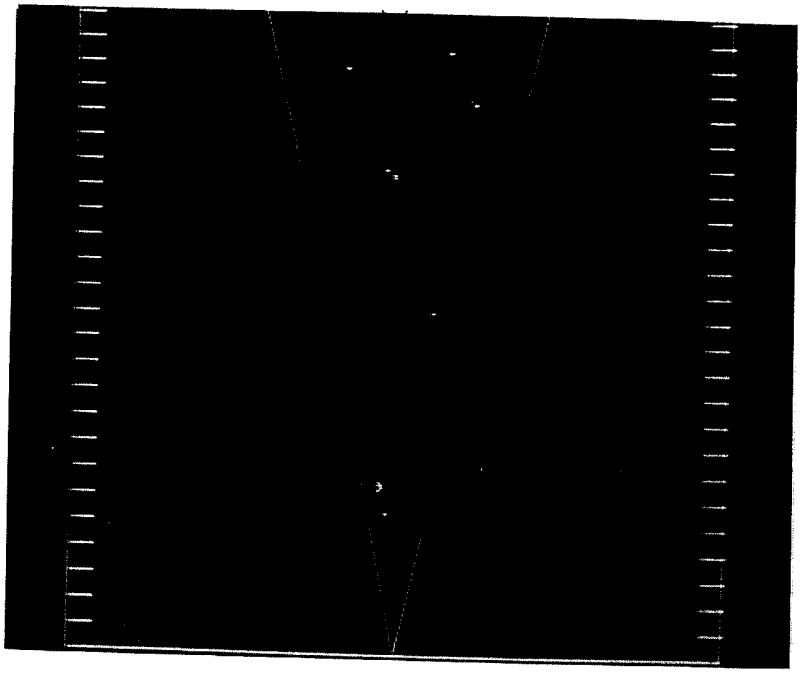
Figure 62: Obstacle extraction

a) Image Row versus depth plot.

b) Image row versus inverse of depth plot.

Least squares line fit to points shown.

(a)



(b)

Figure 63: Extracted obstacles
a) Points on extracted obstacles (thresholded at 30%).
b) Depth to extracted points. The view is of the ground plane from above, with the robot at the bottom center and the V representing the field of view.

be used in restricted circumstances to confirm the position of a real corner as predicted by the line-finding method (Sec. 2.1). This information can then be used for spatial uncertainty management (Chapter 7). A succinct description of the Moravec operator appears in [18] for those readers unfamiliar with its operation.

As the interest operator provides a measure of distinctiveness (how different the pixel region is from its surroundings), the Moravec operator can also be used as a trigger event for spawning **avoid-obstacle** schema instantiations. When distinctive events occur against the relatively unchanging road backdrop, this indicates a potential obstacle. This low-cost focus-of-attention mechanism permits the concentration of higher-cost computational effort in such likely situations.

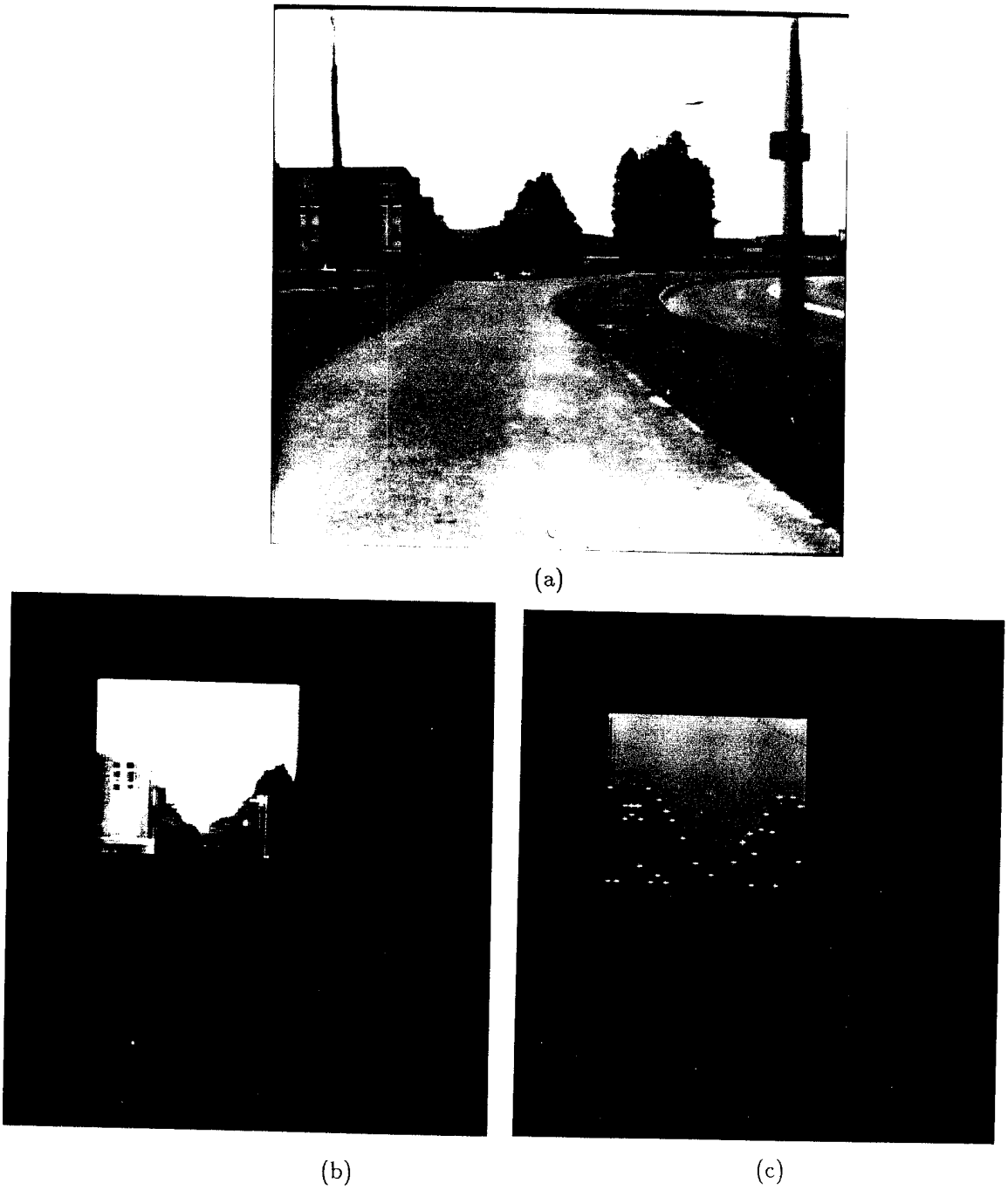
§3. Ultrasonic algorithms

HARV is equipped with a ring of 24 ultrasonic sensors. It should be realized that ultrasonic data is poorly suited for many purposes. Using ultrasound has been likened to "standing in a room completely filled with mirrored objects and having only a penlight glued to your forehead as a source of light: specularity abounds and many surfaces are not visible" [30]. Serious problems involving reflectance and dispersion are present with this sensor modality. Nonetheless, researchers are spending considerable effort trying to utilize ultrasound as a viable means of environmental sensing for mobile robots [e.g. 45,84,43,30]. Drumheller's paper [43] in particular presents an excellent discussion of the limitations for this type of sensor. Having a firm grasp on the problems associated with this data form, ultrasound is used for a limited, although significant, role in AuRA.

HARV's 24 ultrasonic sensors (Polaroid laboratory grade) are controlled by a Z-80 microprocessor and are fired in three banks of eight sensors each to avoid interference. The sonar time-of-flight is converted onboard the vehicle to distances to a surface in tenths of a foot. The limit for detection is 25.5 feet away from our vehicle. No compensation is made for air temperature.

§3.1 *Obstacle avoidance*

The most common mobile robot application for ultrasound is in short-range obstacle avoidance or proximity sensing. For this system, the simplest and most appropriate use



(a)

(b)

(c)

Figure 64: Interest/Distinctiveness (Moravec) operator for localization
a) Original image.
b) Windowed image.
c) Results on finding building corner.

of ultrasonic data is to monitor the ongoing path of the robot and see if vision or any other sensor has missed a path-blocking obstacle. In essence, it serves as a safeguard or fallback system for collision avoidance. This application is fairly straightforward.

In the schema-based system described in Chapter 8, ultrasonic readings are associated with individual motor schemas. If the sensor indicates the presence of an object within the predetermined sphere of influence set by the **avoid-obstacle** schema, a repulsive velocity vector is produced. The magnitude of the vector is determined by the function decreed by the motor schema while the vector's direction is away from the sensed surface. All these vectors, plus any others due to other active motor schemas, are combined by the **move-robot** schema and transmitted to the robot's motor subsystem.

§3.2 *Door finding*

A separate strategy exists which uses ultrasonic data to detect a wall opening and then conduct the robot through it. It works essentially by first locating the wall and moving the robot in a position parallel to it. The robot then maintains an even distance from the wall while following it. The robot monitors for a large sonar reading increase in the direction of the wall coincident with the appearance of a door.

Once the opening is detected, the robot continues moving, waiting for the wall to be detected once again (indicated by a marked drop in the sonar reading). After the wall has been redetected, the robot splits the difference between the two detected wall ends, dead-reckons its position backwards, turns 90 degrees towards the wall and moves through the door.

The schema-based experimental system described in Chapter 8 can execute door entry in a different manner. Repulsive fields generated by the wall itself plus a velocity vector pressuring the robot through the wall replace the hard-coded approach described above. Essentially each sensor that returns a point within the prescribed sphere of influence for an **avoid-static-obstacle** schema generates a velocity vector in the direction away from that point and in magnitude proportional to the closeness of the obstacle. This is counterbalanced by the pressure of the **move-ahead** SI directing the robot to move into the wall. When the door opening appears, the obstacle avoidance pressure lessens and the robot moves into the doorway. As this technique is more consistent with the method of combining primitive motor behaviors to arrive at more complex ones, it is the method

of choice. See Chapter 8 for more information.

§3.3 *Panic sensing*

Ultrasonic sensing is valuable in protecting the robot from immediate danger. Imminent collisions due to obstacles missed by the vision system can cause the system to stop immediately (*freeze*). Moving objects bearing down on the vehicle can cause it to take evasive action (*flee*). The details of both these ultrasonic based processes appears in Section 6.2.

§3.4 *Localization*

Work on localization using sonar (locating the robot's position within the environment) has been progressing. Moravec and Elfes [92,45] perform a matching algorithm on sonar maps to try and position the robot. A variant and extension of this approach is used to supplement visual information for building STM in the AuRA (see Chapter 4). Miller [84] uses path planning that takes into account the positions that are most readily distinguishable for ultrasonic data.

Based on the premise that in a dynamic environment, with unmodeled obstacles possibly appearing and disappearing at any time and at any point in free space, the assumption that the detection of a surface tells us something useful about the robot's global position cannot be made. Ultrasonic data cannot make a qualitative distinction between what types of objects are sensed. It will be assumed that the absence of an expected or predicted surface tells us more than the presence of one. It must be remembered however, that certain critical signal angles must be adhered to if the data is to be considered reliable. If the angle of the signal relative to the wall is too small, the signal may skip off the wall's surface only to return from a more distant point (similar to a bank shot in billiards). Drumheller [43] describes how to limit the sonar data accordingly. By so doing, the uncertainty in the robot's position can be altered (by appropriate modification of the spatial uncertainty map) through the use of ultrasound.

§4. Shaft encoders

Conventional robot systems rely heavily upon the commands given their motors to produce expected changes in the robot's system. The problem with mobile robots is that if a command is given for the robot to move 10 feet, the wheels of the robot will be measured to have rotated the equivalent of a translational movement of 10 feet (by the shaft encoders), but the robot will not necessarily have moved that much. Wheel slippage due to poor traction or uneven tire inflation can produce significant deviations from the commanded movements. The use of terrain modeling can help establish reasonable ranges for errors, but even this is not foolproof. Changes due to a floor being waxed one day and dirty the next, or dewy grass in the morning versus dry lawn in the afternoon can pose serious problems for any *a priori* assumptions made about traversability. Consequently, caution must be used in the interpretation of shaft encoder data. Specific experiments have been performed to determine just how significant the conditions mentioned above are in affecting the translational error (see Chapter 7).

§5. Other desirable sensors

AuRA is an open architecture in the sense that it can readily absorb other sensor modalities. If funds permitted, other sensors could be added to provide greater accuracy or new information for the motor schema system to work with.

An inertial navigational guidance system could supplant the dead-reckoning system of the robot. Inertial guidance measures the actual rotations and translations of the vehicle as opposed to shaft encoders counting the number of wheel turns. The significance of this in limiting uncertainty and improving landmark prediction cannot be overemphasized. Unfortunately such a system is extremely expensive.

Inclinometers are desirable for several reasons. Information on topography could be obtained and correlated against an extended meadow-map representation of the world. Motor schemas for literal "hill-climbing" could be built (i.e. **move-up, move-down, stay-on-even-keel**). Registration of image sequences could be aided by knowing the actual differences in roll and pitch between frames. Distortions in expected landmark positions could be foretold by measurement of the camera's tilt. Inclinometers are not

overly expensive and are a suitable short-term goal for addition to AuRA.

A laser scanner for providing depth to environmental surfaces is highly desirable. This active sensor could supplant the computationally expensive and currently fragile depth-from-motion algorithm. Once again, however, cost is a factor. Several research groups are working with laser scanners nevertheless [141,94] and perhaps in the near future these sensors will be more affordable.

§6. Perception subsystem

Several disjoint topics all related to the perception subsystem constitute this section. Some are related to the general AuRA architecture (Fig. 2), others are more specifically concerned with the first pass implementation (Fig. 3). These topics include sensor processes, panic processes, and camera calibration.

§6.1 *Sensor processes*

Sensor processes serve as the gateway to both the clipboards described in Chapter 3 for the first pass implementation (Fig. 3) or the VISIONS system in AuRA's more general form. The role of sensor processing is to convert the raw data into a form that is readily integrated into the available representations (e.g. STM or schemas). The number of turns of a motor shaft or the time of flight of an ultrasonic return is of little value without some preprocessing.

Sensor processing in the case of vision involves the digitization of the incoming video image from the camera. This is currently performed on a Gould IP8500. Image resolution is also reduced from 512 by 480 pixels to a 256 by 256 pixel image. In some applications it may also be desirable to temporally average multiple frames to minimize the effects of noise, or to perform some form of smoothing prior to the image being posted on the clipboard.

Ultrasonic sensor processing utilizes the Z-80 microprocessor onboard the DRV to convert the time of flight of the ultrasonic return to a distance measured in tenths of feet. The hardware and firmware for this task were provided by Denning Mobile Robotics with the vehicle. Although, it does not take into account air temperature and other factors which can affect the result, the values returned seem adequate for the purposes they are

used for. The results are reported for each of the 24 sensors in tenths of a foot. The firing sequencing is also controlled to prevent overlap of return signals and resultant false readings.

Finally the DRV, again using another separate dedicated Z-80 microprocessor, converts the shaft encoder data into a form that is easier to interpret than simply the number of motor revolutions. Cartesian X and Y values representing the "distance" traveled (based on the number of drive motor shaft revolutions) is reported to the clipboard. Information regarding the direction the robot is facing is also available based on the steering motor encoder data. From the uncertainty treatment described in Chapter 7, it should be obvious that the "distance" the shafts have rotated does not correspond accurately with the actual location of the robot. Nonetheless it does provide valuable data which are used to constrain the spatial uncertainty map.

§6.2 *Panic processes*

Panic processes serve to alert the robot in a reflexive manner to potentially dangerous situations. The warnings and resultant commands issued by these processes bypass higher level processing and are communicated directly to the vehicle interface. The time saved in emergency situations can be critical to a successful response.

Several types of panic processes are conceivable. Most have animal behavior parallels. These include "freezing" in place in response to some unexpected event or "fight or flight" behaviors based on the approach of another entity. For the first pass AuRA implementation, the panic-stop ("freezing") and panic-avoid ("flight") behaviors have been constructed using ultrasonic data.

The panic-stop process continuously monitors the data posted on the clipboard by the ultrasonic sensor process. If any reading in the direction of the robot's motion is below some predetermined threshold (e.g. 2 ft), a continuous stream of stop commands is sent to the vehicle interface. This immediately prevents the robot from moving by counteracting any old motor commands currently being executed and preventing the implementation of any new motion commands. The panic-stop process continues to monitor the incoming sonar data and only when the offending obstacle is removed as evidenced by safe sonar readings does it allow the robot to continue moving. Information is also posted on the clipboard for other processes (such as the navigator and motor schema manager) to react

to the blockage in a more timely manner.

The panic-avoid process constantly polls incoming ultrasonic sensor data to see if an object is approaching the robot in a head-on direction. It accomplishes this by comparing successive time-stamped sonar readings from the clipboard, noting the difference in distance when compared to the robot's velocity, and determines if the robot is being approached. When a predetermined threshold for time-to-contact is reached, the robot turns 90 degrees and moves rapidly to the side until it no longer senses the object approaching. This is an evasive action maneuver. The panic-stop process remains in play preventing it from crashing into a wall or other object. The panic motor commands override any other commands already executing or waiting to be executed within the robot. Several consecutive confirming sonar readings are required to trigger the panic-avoid process, minimizing the likelihood of spurious ultrasonic data causing this event. The result is somewhat comical in appearance, but it is potentially very useful behavior for a mobile robot operating among moving equipment. In those cases, it is not enough for the robot to stop. It must get out of the way or it may possibly be damaged by (or damage) the moving body. Once the panic-avoid episode is completed, special procedures are invoked to allow the robot to regain its bearings and then satisfy the previously specified navigational goals.

The major drawback to the current implementation of the panic processes is the slow transfer rate for the ultrasonic data over the serial line to the VAX. A future solution would be to embed these routines onboard the DRV's MC68000 processor eliminating the communications delay to the VAX. The analogy to reflex arc behavior becomes even more apparent in such a circumstance.

§6.3 *Camera calibration*

Much work has been performed in the area of camera calibration, making the solution to the problem fairly straightforward. Monocular video systems are particularly well understood. Kak [62] and Thompson [123] both present excellent descriptions of the mathematics and techniques required for camera calibration. Stereo camera calibration has been discussed by many authors including [93,70]. Even a trinocular video system calibration has been described [72].

The mobile robot project was fortunate to receive camera calibration software from

the University of Rochester. Rigoutsos describes the process and constraints for this work in [111,112,113]. The code has been slightly modified for use in the calibration of the camera mounted on the vehicle.

A known three-dimensional Cartesian map is made of several readily recognizable points in our hallway, such as door corners, light fixtures, and hall borders. (Fig. 65 and Table 2). The robot is then rolled to a position in the hall. An image is taken. The image coordinates for each real world point are found (currently 20 points are used). Rigoutsos' algorithm operates on this data, producing the 4 by 3 calibration matrix. This matrix is used to convert points whose position is known *a priori* in (x,y,z) space relative to the robot to their expected image (u,v) coordinates. The prime user of this matrix is the Expecter process in the uncertainty management subsystem.

§7. Summary

Perceptual strategies are embedded in motor schemas to provide the necessary information for the robot to interact intelligently with its world. These strategies can take many forms but are based on the premise of action-oriented perception. This concept allows special-purpose perceptual techniques to be concentrated on individual components of the navigational process. These include methods for the detection of obstacles, pathways and landmarks.

The vision algorithms used in AuRA encompass a wide range of computer vision techniques. These include primitive interest operators, more sophisticated line-finding and region segmentation algorithms, a multiple frame depth-from-motion algorithm, and potentially, a scene interpretation system. Each approach has its purpose, advantages and disadvantages for use in mobile robot navigation. In all cases, however, versions of vision algorithms have been developed which extract image features rapidly (at the expense of reliability in some cases). In addition, the control of all algorithms attempts to use a top-down strategy of restricting processing to windows based upon LTM or previous frames. In this manner, real-time processing may be achieved for certain interesting navigation tasks that might not have been feasible until more powerful parallel hardware arrives.

Some of the specific contributions made in this chapter involve the adaptation of vision algorithms (FLF, FRS, depth-from-motion, interest operator) to mobile robot

Figure 65: Camera Calibration Scene

Table 2: Camera calibration data

Point	X	Y	Z	U	V	Description
1	2.125	86.0	374.125	87	391	Corner of door
2	100.00	14.0	393.0	395	152	Top of outlet
3	102.063	82.0	529.00	361	337	Tag on A207
4	0.000	37.5	782.00	171	236	Door knob
5	52.5	46.75	1215.00	251	245	Bar on right door
6	80.5	95.25	366.875	348	422	Corner of tile
7	92.375	6.0625	535.625	340	162	Corner of strip
8	73.5625	93.125	352.875	328	423	Box on ceiling
9	73.25	92.875	473.375	307	386	Box on ceiling
10	2.00	5.9375	747.0	167	182	Frame of door
11	2.00	5.875	298.375	40	100	Door frame
12	32.75	95.75	380.25	192	415	Tile w/metal frame
13	51.00	10.375	1217.25	248	210	Corner of door
14	92.125	6.0625	1063.875	296	201	Corner of strip
15	102.5	6.5	271.5	423	88	Corner of strip
16	2.25	86.0	298.5	49	425	Corner of door
17	2.00	6.25	374.5	85	131	Black strip
18	2.5	83.5	747.25	168	308	Corner of door
19	44.0	95.5	1205.5	243	294	Corner of exit sign
20	5.75	68.75	618.5	149	294	Corner of sign

Calibration matrix

$$\begin{vmatrix} 5.196073 & -0.912248 & 0.946008 & -130.762170 \\ -0.829986 & 4.604287 & 0.879309 & -25.105374 \\ -0.003328 & -0.005164 & 0.003582 & 1.000000 \end{vmatrix}$$

navigation. This necessitated the development of representations (LTM) suitable for providing expectations for these algorithms. Application of the FLF and FRS algorithms to actual image sequences enabled path following behavior to be undertaken (Chapter 8). The actual perceptual algorithms are only a part of the overall system (in some cases a small part) and additional code involving the production of expectations, image sequencing, vehicle servoing, communications, etc., had to be produced before viable experiments could be undertaken. It's a big step from an image on a video monitor to intelligent motor action.

The ability of each of the vision algorithms to function in a domain as unconstrained as outdoor navigation varies considerably. The fast line finding algorithm's path-following software is the most robust. The ability to work with highly fragmented lines even under conditions of partial occlusion of the road edges makes it very versatile. Its capacity to be tuned based on expectations from either previous images or knowledge from LTM, couples it tightly with AuRA's design philosophy of action-oriented perception.

The fast region segmentation algorithm holds great potential when it is extended into the color spectrum. HARV only has the capability to digitize monochrome images currently, but that will change shortly. Then full advantage of the spectral characteristics of environmental objects will be exploited. Nevertheless, even in its current form it can be applied to useful tasks such as path following (Chapter 8).

The depth-from-motion system is a major project that is receiving considerable research effort at the University of Massachusetts. Preliminary results regarding frame-to-frame point tracking are very promising, but the difficulties remaining in FOE extraction and image registration need to be solved in order to produce a robust passive navigation system. Preliminary results using HARV as the test vehicle for this algorithm are presented in Chapter 8.

Other sensor forms are used within AuRA, taking advantage of their distinguishing characteristics. Ultrasonic data is well suited for obstacle avoidance and can serve as a confirmation mechanism for landmark recognition. The processing speeds for ultrasonic data make it suitable for conducting real-time continuous motion navigation with HARV (Chapter 8). As limited as this sensor modality is, it can still be used quite advantageously. Shaft encoder data provide constraints on the limits of motion that the robot has undertaken (Chapter 7). Additionally, they provide estimates of where a navigational

goal extracted from LTM is located relative to the robot's current position. As newer and better sensors become available, AuRA's design will easily accommodate them, extending this system's capabiilties even further.

A major drawback of the vision algorithms is the time required to process them. Although low resolution images may be suitable for industrial robotic applications, natural scenes require more detail. A 256 by 256 image contains a very large quantity of data. The time required to process such an image on our existing hardware forced us to use "lurch" mode for the visual navigation experiments. Continuous motion was possible, however, using ultrasonic data. Chapter 8 describes the experiments that were performed using both vision and ultrasound with our mobile robot HARV.