Robot Navigation Using Image Sequences

Christopher Rasmussen and **Gregory D. Hager**

Department of Computer Science, Yale University
51 Prospect Street
New Haven, CT 06520-8285
rasmuss@powered.cs.yale.edu, hager@cs.yale.edu

Abstract

We describe a framework for robot navigation that exploits the continuity of image sequences. Tracked visual features both guide the robot and provide predictive information about subsequent features to track. Our hypothesis is that image-based techniques will allow accurate motion without a precise geometric model of the world, while using predictive information will add speed and robustness.

A basic component of our framework is called a *scene*, which is the set of image features stable over some segment of motion. When the scene changes, it is appended to a stored sequence. As the robot moves, correspondences and dissimilarities between current, remembered, and expected scenes provide cues to join and split scene sequences, forming a map-like directed graph. Visual servoing on features in succcssive scenes is used to traverse a path between robot and goal map locations.

In our framework, a human guide serves as a scene recognition oracle during a map-learning phase; thereafter, assuming a known starting position, the robot can independently determine its location without general scene recognition ability. A prototype implementation of this framework uses as features color patches, sum-of-squared differences (SSD) subimages, or image projections of rectangles.

Introduction

A robot whose duties include navigation may be called upon to perform them in many different environments. In such fluid situations, it is unlikely that a geometric model will always be available, and arranging and testing fiducial markers may be onerous or impossible. An attractive way to alter the job description of a navigating robot would be to simply "show" it the path(s) it should follow, and have it rely on its own sensing systems to develop an internal representation sufficient to allow repetition of the path within some bounds. Specific points of interest should be easy to indicate, although special locations (*e.g.* a delivery site) may be adorned with guide markers which can be used for precise position control once they are approached. Our aim is to develop a vision-based navigation system capable of such performance. Our choice of vision is based on the necessity of observing large areas to find useful landmarks for defining the path, and the need to distinguish them from one another as easily as possible.

Navigation depends on the notion of location, which can be characterized in a number of ways. Strictly metrical approaches describe it as a position in a fixed world coordinate system. The system in (Kosaka & Kak 1992) knows the environment's three-dimensional geometry and its initial position and orientation; the navigation task is to find a path to a goal position and orientation. Kalman filtering is used to reconstruct and predict the coordinates of model features, which are matched at regular intervals during motion. The sonar-based algorithm of (Leonard, Durrant-Whyte, & Cox 1990) builds its own map of beacons' absolute positions using Kalman filtering to account for uncertainty. Their mapping process is dynamic in that the robot constantly compares sensory evidence with expectations, and updates its map and/or position estimate when the two disagree. Prediction allows directed sensing: "knowing where to look" can improve the speed and reliability of the mapping process.

Graph-based techniques smear out exactness somewhat by relating position regions to nodes or "places" in a connectivity graph, and rely on closed-loop feedback to guide the robot from place to place. In (Taylor & Kriegman 1994) the model of the environment is a graph constructed while exploring, based on visibility of barcode landmarks. Similarly, in (Kortenkamp et al. 1992), (Kuipers & Byun 1981), and (Engelson 1994), the robot learns a model of the environment by building a graph based on topological connectivity of intercsting locations, but it attempts to decide what constitute landmarks on its own. The robot in (Kortenkamp et al. 1992) learns the environment's topol-

ogy from doorways detected by sonar while storing images for place recognition. The robot in (Kuipers & Byun 1981) uses sonar to detect places that maximize a distinctiveness function and to match or add them to its map. A disadvantage of graph-based approaches is their tendency to rely on a solution to the difficult problem of general place recognition.

We avoid reference to absolute coordinates by taking advantage of vision to define locations by the notion of view. A robot is at place A if what it sees corresponds to what can be seen at place A. Given that we know where we are, we use visual tracking to extend the definition of place to a range of motions, and require that all (or nearly all) of the world corresponds to some place. We then utilize the continuity of our representation to predict changes of view between places, thereby eliminating the need for a strong notion of recognition. Navigation then becomes a problem of moving from view to view. The desired motion is computed using visual feedback between what the robot currently sees, and what it saw when it was at the goal location before.

The work described in (Fennema et al. 1990) is similar to ours. They use a servoing approach to navigation, but with reference to an accurate geometric model of the environment which we do not assume. Navigation proceeds by identifying landmarks along a path through the graph that are visible from their predecessors, and servoing on them in succession via visual feedback. Recognizing landmarks is a matching problem between 2-D image data and the 3-D world model. Other route-based approaches to navigation are described in (Hong et al. 1992) and (Zheng & Tsuji 1992). The robots in these papers are guided in a learning stage through the environment, periodically storing whole panoramic images. A graph is built up by matching new images with the known corpus; paths can be planned in the completed graph with a standard algorithm. When executing a path, image-based servoing is used to traverse individual edge segments. (Kortenkamp et al. 1992) argue that taking snapshots at regular intervals leads to the storage of much unnecessary information. Moreover, new scenes must be matched against all remembered ones, an inefficient process. Our system addresses the first of these issues by finding image features and storing them only as frequently as necessary for servoing. We sidestep the place recognition problem with human assistance during a learning phase; general recognition ability is not needed during navigation.

Map Construction

In order to present our navigation system it is useful to first define some terminology and to sketch the procedure used to create the internal map used by the robot.

Terms

We use the term *marker* to denote any visual entity that is in some way distinctive or meaningful to the system. A marker is characterized by a set of relatively invariant properties (color, shape, texture, function) and a dynamic state (image location and size). Within most images there are many potential markers that can be discovered and analyzed; the current *scene* is the set of markers upon which the robot is currently focusing its attention. The question of what kind of visual entities will gain attention is left to the implementation, but these can run the gamut of complexity from chairs, doors, and trees to lines, points, and splotches of color. The limiting factors of the recognition process are sophistication and speed; these must be traded against the distinctiveness, trackability, and rarity of the markers thus found.

As an additional consideration, to successfully servo from one scene to another certain geometric constraints on the number and configuration of the markers in them must be met. These constraints vary according to what kinds of markers are used. As a shorthand, we will say that an evaluation function $C(s)$ returns a truth value T if scene s meets these criteria and F otherwise. Likewise, we shall say that two scenes, s_1 and s_2 are *equivalent*, $s_1 \equiv s_2$, if an abstract comparison function returns true. Intuitively, equivalency is measured by some matching function which is invariant over small changes in viewpoint caused by the motion of the robot. Furthermore, we say that the robot is viewing a previously stored scene s_i if $s_{viewed} \equiv s_i$, where s_{viewed} is the scene currently being viewed.

A *sequence* is an ordered list of scenes stored as the viewer moves through an environment. The currently-viewed scene is added to the current sequence when there is a *scene change*: here we use the occurrence of a marker appearing or disappearing to signal this. When a marker appears, the current scene is added. When a marker disappears, the last scene containing the disappearing marker is added. In essence, scenes serve as key frames for reconstructing movement later through visual servoing (Hutchinson, Hager, & Corke 1995).

A *location* is a set of two or more scenes from different sequences that are the same under the equivalency function; this is where sequences intersect. There are two kinds of locations: *divergences* and *convergences*.

A divergence is a choice point: multiple possible sequences can be followed away from a single scene. Convergences are scenes at which multiple sequences terminate. Since a scene can satisfy both of these definitions (where two corridors cross each other, for instance), the type assigned to a location can vary according to its place in the order of scenes encountered by the robot. The key quality of locations is that they are topologically meaningful events that can be recognized by the robot simply by keeping track of what it is seeing and what it remembers. A *map* is the directed graph consisting of all sequences recorded and all locations noted by the robot as it moves.

Map-Building

The state of the robot as it makes a map is described by the following variables: $\{S\}$, the set of sequences recorded so far; $\{L\}$, the set of known locations; S_{new}, the sequence the robot is in; $\{s_{possible}\}$, the set of scenes that the robot could be viewing; and s_{viewed}. s_i^* denotes the set of scenes reachable in one scene-step from s_i via the current map graph.Map-making begins with $\{S\} = \emptyset$; $\{L\} = \emptyset$; $S_{new} = ()$, the empty list; and $\{s_{possible}\} = \infty$, the set of all possible scenes. This corresponds to seeing *unfamiliar* scenery; *familiar* scenery is indicated when $\{s_{possible}\}$ has a finite number of members. When the robot is instructed to cease mapping, s_{viewed} is added to S_{new} and exploration halts. It is assumed that an exploration process that drives movement is running in the background. This could be the human guide suggested, a sonar-based obstacle avoidance routine, or any other desired program. While not finished, s_{viewed} is updated continually until $C(s_{viewed}) = F$; whenever s_{viewed} changes the steps in Figure 1 are executed.

The key question in this algorithm is the nature of the scene comparison function, with the additional problem of efficient image indexing to determine the confluence of sequences. We describe the current instantiation of equivalency in the Implementation section. For the moment, we rely on a human guide to inform the robot when a convergence or divergence has occurred, and hence do not need to solve the general indexing problem.

In Figure 2 the map-making process is illustrated for a sample environment. By continuing this process of exploration, the robot would eventually arrive at a complete map of the floor, but even in its intermediate forms it has partial usefulness. A more complete exposition is given in (Rasmussen & Hager 1996).

1. If unfamiliar, does $\exists\ s_i \in S_j = (s_0, \ldots, s_n)$ s.t. $s_{viewed} \equiv s_i$?
   (a) Yes (convergence):
      i. Add location $\{s_{viewed}, s_i\}$ to $\{L\}$
      ii. If $S_{new} \neq S_j$, then replace S_j in $\{S\}$ with $S_{j_1} = (s_0, \ldots, s_i)$ and $S_{j_2} = (s_i, \ldots, s_n)$
      iii. Set $\{s_{possible}\} = s_i^*$
   (b) No: Append s_{viewed} to S_{new} in $\{S\}$
2. If familiar, does $\exists\ s_i \in \{s_{possible}\}$ s.t. $s_{viewed} \equiv s_i$?
   (a) Yes: Set $\{s_{possible}\} = s_i^*$
   (b) No (divergence):
      i. Add $S_{new_1} = (s_0, \ldots, s_i)$, $S_{new_2} = (s_i, \ldots, s_n)$ to $\{S\}$, and add location $\{s_i\}$ to $\{L\}$
      ii. Replace S_{new} in $\{S\}$ with $S_{new} = (s_i, s_{viewed})$
      iii. Set $\{s_{possible}\} = \infty$

Figure 1: Decision process for map-building

Using the Map

During and after map-building, the robot can navigate between any two scenes in the map that are connected by a *traversable* path. A traversable path is a series of sequences or partial sequences in the map, all directed head to tail, from the robot's current scene to the goal scene. As the learning process progresses, more places in the environment will become represented in the map by scenes, and more pairs of scenes will be traversable. In order to initialize this process, the robot must analyze the current scene, identifying markers, and then match them to the markers it expects to see given its current location.

The basic building block of navigation is the *traversal*. Let s_i and s_j be scenes in the map such that the robot is viewing s_i and there is a traversable path from s_i to s_j. Let $M_{ij} = s_i \cap s_j$ and define s_j' to be s_j restricted to the markers in M_{ij}. If M_{ij} is not empty and $C(M_{ij}) = T$ (always the case if $j = i + 1$), then by using some variant of visual servoing the robot can move until $s_{viewed} = s_j'$ within an arbitrary tolerance. In the limit, this movement will bring the robot to the absolute position and angle that it was at when it recorded s_j.

In many cases, M_{ij} is empty, meaning that the goal scene is not even partially visible from the initial scene. This necessitates traversing intermediate scenes and performing a *handoff* between each. Let s_i, s_j, and s_k be scenes in the map such that the robot is currently s_i and there is a traversable path from s_i to s_k through s_j. Let M_{ij} and M_{jk} be defined as above such that both are non-empty, but $M_{ik} = M_{ij} \cap M_{jk}$ is empty. The robot cannot traverse directly from s_i to s_k; instead it must first servo to s_j using markers in M_{ij}

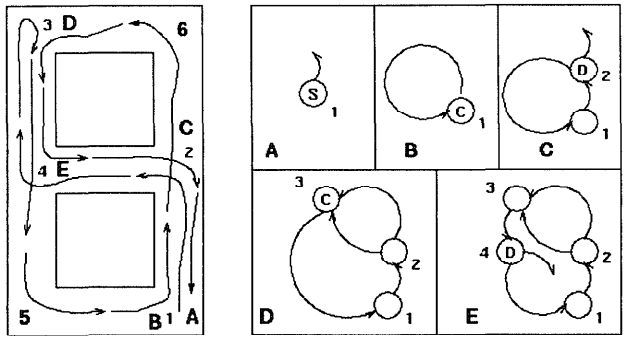


Figure 2: Hypothetical exploration. The path the robot follows is shown schematically on the left, starting at 1 (numbers are for spatial reference only); successive updates of the map occurring at steps A-E are shown on the right. Location nodes are marked to indicate why they were created: S = start event, C = convergence, and D = divergence.

until the markers in M_{jk} are found, and then "hand off" and servo on those markers to the destination.

The simplest way of implementing a handoff is to rely on the default marker search technique used for mapping. As the robot servos toward s_j and searches for new markers concurrently, it may find all of M_{jk} before it ceases movement, allowing it to smoothly switch servoing control at the appropriate moment. If not all of M_{jk} is found before stopping, the robot can continue searching until it acquires the markers, then change control to the next servoing phase. In practice, though, the time needed to find the markers can be quite long, and the matching process is likely to be error-prone. Furthermore, if a significant location discrepancy exists between s_{viewed} and s'_j at the end of servoing, not all of the desired features for s'_k may be in view. Angular errors are especially significant because they often cause more distant markers in M_{jk} to be partially or completely out of view.

Correcting such errors is critical, so we have investigated a predictive mode for marker location in which the robot dynamically estimates the image locations of M_{jk} as it moves (Rasmussen & Hager 1996). Since we do not have an underlying geometric map, all feature prediction must operate directly from image information. Thus, we have adapted a result from the theory of geometric invariance (Barrett et al. 1992) to vehicles moving in a plane. We have found that when the image projections of four three-dimensional points p_0, p_1, p_2, p_3 are known in two different views v_1 and v_2, knowledge of the image projections of p_0, p_1, and p_2 in a third view v_3 is sufficient to predict the image location of p_3 modulo certain geometric constraints on the points and the transformations between views.

This result means that if three points in M_{ij} and one in M_{jk} are simultaneously visible in two frames during map-building, then during servoing on the markers in M_{ij}, any points that were visible in M_{jk} have predictable locations. The predictions are somewhat noisy, but they are reasonable hints for initializing and limiting search procedures. Furthermore, if the predicted point locations fall consistently outside the image, the robot can make an open-loop adjustment to bring them into view.

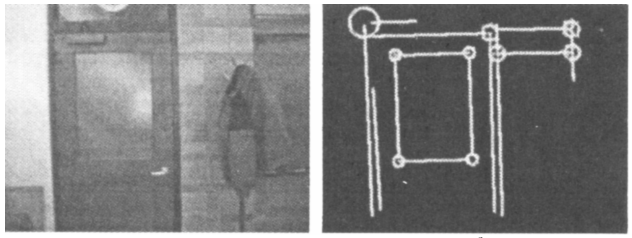
Putting this together, a navigational directive is issued as a pair of scenes (s_{start}, s_{finish}), where the robot is assumed to be currently viewing s_{start}. A path-planning algorithm returns an ordered list of scenes constituting a traversable path to the goal. This list is input to a greedy scheduling algorithm that selects which set of markers can be servoed on for the most consecutive scenes before the robot must switch to a new target set. This approach is based on a desire to minimize the number of handoffs.

Implementation

We have implemented most of the architecture described above on a tethered Nomad 200 robot (lack of an onboard digitizer has limited movement to one lab and an adjoining hallway). In this section, we describe the implementation of various system modules.

3-D Rectangles

One type of marker we have tested is image rectangles, defined by two approximately horizontal image edges and two approximately vertical image edges at least 20 pixels in length meeting at four distinct corners. We are using the XVision system (Hager 1995) for real-time visual tracking as the basis for our marker identification and tracking algorithms, which are described in detail in (Rasmussen & Hager 1996). Rectangles were chosen primarily because they are simple to detect and track, and in an indoor environment they are relatively common. The default marker-finding method is random search over the entire image. When traversing a previously known area, the same search procedure is used, but the markers found are matched to those previously located in the scene. For our limited experiments, we have found matching rectangles by aspect ratio alone satisfactory, although we have also experimented with using statistics on the gray value distribution of the interior of the rectangle as well. Based on the techniques described in (Chaumette, Rives, & Espian 1991), the four corners of one rectangle are sufficient to servo on. Thus, the scene evaluation function $C(s)$ is simply that $s \neq \emptyset$. To maintain this condition, the robot must block movement whenever it loses track



a b

Figure 3: Rectangle-finding. Corners are marked with circles of radius inversely proportional to the confidence of correctness. The door window was the only rectangle found here.

of the only marker it has until it finds another one.

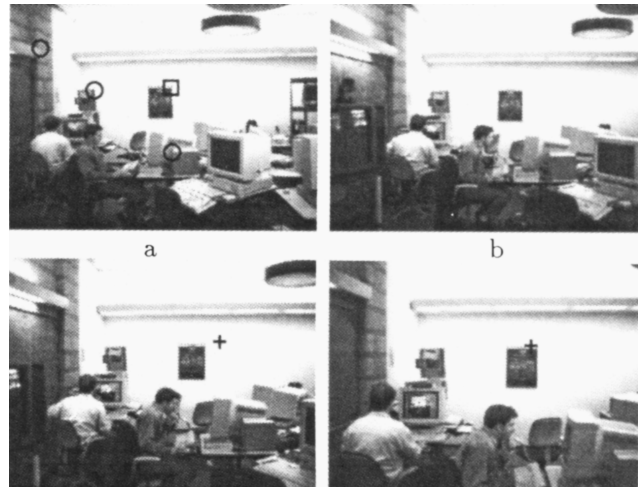
Experimental Results

We have tested the above modules on a variety of indoor scenes. This section briefly describes our experimental results.

Marker Finding When not using prediction, the performance of our rectangle-finding algorithm varied. In particular, it performed poorly on cluttered scenes, due to the combinatorics of testing many pairs of edges as possible corners and many groups of corners as possibly belonging to rectangles. It performed roughly as well at locating lines when moving slowly as when immobile. A sample image and the rectangle found in it is shown in Figure 3.

One central issue is the ability of the algorithm to find markers quickly, and to consistently find roughly the same set of markers in an image. On a SPARC 10/40, over twenty runs of the algorithm on the door view in Figure 3, it found the window rectangle 11 times within 60 seconds (for those times, in an average of 29 seconds). Four other times it found another rectangle in the tile pattern on the wall above the coat rack. In the other five trials it did not find any rectangle in less than 60 seconds. Slight variations in robot position and viewing angle did not affect performance.

As described above, our goal is to use prediction to improve performance when attempting to refind markers. Searching the entire 640 x 480 image is cumbersome and slow. Our view is that prediction provides a seed about which a local search can be centered. If we are looking specifically for the door window rectangle of Figure 3.a and assume that prediction has yielded a seed approximately in its center, we can constrain search to a smaller window tailored to the remembered area of the rectangle. Accordingly, another 20 runs were done with random search limited to a 300 x 300 square uniform distribution about the door window. The door window rectangle and only the window rectangle was found for all 20 trials. Furthermore, the average time to find it was reduced to only 12 seconds. In this manner distracting lines elsewhere in the image were effectively ignored.



a b c d

Figure 4: Prediction. (a) and (b) are calibration views taken at the start and finish of a three-foot leftward translation of the robot, with four tracked SSD features marked. (c) and (d) are the first and last images in a sequence of six taken at equal intervals along a ten-foot forward translation from (b). In them, only the image location of the feature marked with a square in (a) is unknown to the robot. The crosses indicate the predicted locations of this feature.

Prediction In Figure 4 the performance of our prediction scheme is illustrated. We must imagine that the first two images were taken during learning, and the other two (along with the four not pictured) were taken later as the robot consulted its map. The predicted feature is one that it wants to pick up while moving in order to effect a smooth handoff.The mean error magnitude over the six images of the sequence between the predicted and actual x coordinate of the feature was 14.1 pixels, and for the y coordinate it was 2.1 pixels; yielding a mean distance error of about 14.3 pixels. This is close enough that the robot could find it very quickly with a local search.

Servoing Traversal-type servoing has been successfully done in a number of situations (such as approaching the door in Figure 3). We are working to improve the reliability of handoffs by addressing the angular error problem described above.

Discussion

One vulnerability of the scene-based framework is its assumption that an acceptable group of markers will always be in view. For any kind of markers chosen, in

a real-world environment there will be gaps. In the indoor environment expected by our implementation, for instance, there are not enough closely-spaced rectangles for the robot to handoff its way around a ninety-degree corner. Even straight corridors may be relatively featureless for long stretches, rendering them impassable. Clearly, we can use odometry locally as a second source of predictive information. Open-loop movements based on saved odometric information could be useful on a limited basis—say, to negotiate corners. The system we have presented is fairly robust with respect to slight angular and positional errors, and errors would not be cumulative.

Another possible solution is to consider a different kind of servoing. The servoing we have focused on so far is *transformational*, insofar as the robot tries, by moving, to make a tracked object change appearance to look like a target object. A different, *homeostatic* paradigm is suggested by road-following and wall-following systems (Turk et al. 1988). In them, the goal is not to change one sensory entity into another through movement, but rather to move while maintaining a certain constancy of sensory input. Road-followers seek to move forward while keeping the sides of the road centered. Something like this could be used to guide movement while no target is visible. If the robot can discern situations in which the maintenance of a visual constraint—e.g., staying centered in a hallway, keeping a building in front of it, etc.—can provide appropriate cues for movement, then it can bridge gaps between homing-type targets. The difficulty of this approach is that the robot does not know accurately, as with our framework, when something should be visible. Rather, it must be vigilant for the appearance of a marker signalling that it has arrived. Some odometry, either in distance traveled or time spent, would help limit the time it spends looking for such a signal.

With multiple markers in one scene, joining these individual discrimination criteria to constraints imposed by spatial relationships (above, next-to, etc.) as well as the invariant relationship used for prediction would lead to still greater confidence in matching. Finally, an important area of additional research is dynamically updating the map during motion. In particular, it is likely that certain markers are stable and easy to find, while others occasionally appear and disappear. Ideally, as the robot moves around it should modify its internal representation to more heavily weight more reliable markers, and possibly add new markers.

In conclusion, we have presented a vision-based navigation framework for mobile robots that requires no *a priori* environmental model and very little odometry, and implemented most of its fundamental components. The notion of the environment that it builds remains on the image level, both as stored and when generating movement. It is also modular, permitting easy modification, especially to its recognition and matching capabilities and its task description.

Acknowledgments

This research was supported by ARPA grant N00014-93-1-1235, Army DURIP grant DAAH04-95-1-0058, by NSF grant IRI-9420982, and Yale University.

References

Barrett, E.; Brill, M.; Haag, N.; and Payton, P. 1992. Invariant Linear Methods in Photogrammetry and Model-matching. In *Geometric Invariance in Computer Vision*, Mundy, J., and Zisserman, A. eds., Cambridge, Mass.: MIT Press.

Chaumette, F.; Rives, P.; and Espian, B. 1991. Positioning of a Robot with respect to an Object, Tracking it, and Estimating its Velocity by Visual Servoing. In Proc. IEEE Inter. Conf. Robotics and Automation, 2248-53. Sacramento, CA, April.

Engelson, S. P. 1994. Passive Map Learning and Visual Place Recognition. Ph.D. diss., Dept. of Comp. Sci., Yale Univ.

Fennema, C.; Hanson, A.; Riseman, E.; Beveridge, J.; and Kumar, R. 1990. Model-Directed Mobile Robot Navigation. *IEEE Trans. Systems, Man, and Cybernetics* 20(6): 1352-69.

Hager, G. 1995. The 'X-Vision' System: A General-Purpose Substrate for Vision-Based Robotics. *Workshop on Vision for Robotics.*

Hong, J.; Tan, X.; Pinette, B.; Weiss, R.; and Riseman, E. 1992. Image-Based Homing. *IEEE Control Systems*: 38-44.

Hutchinson, S.; Hager, G.; and Corke, P. 1995. A Tutorial Introduction to Visual Servo Control. *IEEE Trans. Robotics and Automation.*

Kortenkamp, D.; Weymouth, T.; Chown, E.; and Kaplan, S. 1992. A scene-based, multi-level representation for mobile robot spatial mapping and navigation. Technical Report, CSE-TR-119-92, Univ. of Michigan.

Kosaka, A., and Kak, A. C. 1992. Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties. *CVGIP: Image Understanding* 56(3): 271-329.

Kuipers, B., and Byun, Y. T. 1981. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems* 8: 47-63.

Leonard, J.; Durrant-Whyte, H.; and Cox, I. 1990. Dynamic Map Building for an Autonomous Mobile Robot. *IEEE Inter. Workshop on Intelligent Robots and Systems*: 89-95.

Rasmussen, C., and Hager, G. 1996. Robot Navigation Using Image Sequences. Technical Report, DCS-TR-1103, Yale Univ.

Taylor, C., and Kriegman, D. 1994. Vision-Based Motion Planning and Exploration Algorithms for Mobile Robots. *Workshop on the Algorithmic Foundations of Robotics.*

Turk, M.; Morgenthaler, D.; Gremban, K.; and Marra, M. 1988. VITS—A Vision System for Autonomous Land Vehicle Navigation. *IEEE Trans. Pattern Analysis and Machine Intelligence* 10(3): 342-61.

Zheng, Z., and Tsuji, S. 1992. Panoramic Representation for Route Recognition by a Mobile Robot. *Inter. Journal of Computer Vision* 9(1): 55-76.