Sketch-Based Navigation for Mobile Robots

George Chronis and Marjorie Skubic
Computer Engineering and Computer Science
University of Missouri-Columbia

***Abstract*- The goal of this work is to create a robot interface that allows a novice user to guide, control, and/or program a mobile robot to perform some task. To illustrate our interface we have chosen the example of robot navigation. We present a way to use a PDA computer to sketch a map and a robot route on this map. Qualitative instructions are then extracted from such a map to form a list of sequential steps that the robot has to follow to complete its task based on landmark states. We also demonstrate the feasibility of our approach on a Nomad200 robot simulator.**

I. INTRODUCTION

The goal of this work is to create an interface that allows a novice user to guide, control, and/or program a semi-autonomous robotic assistant to perform some task.

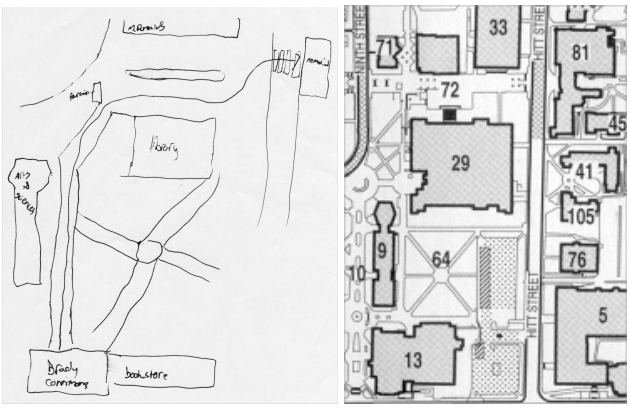
As one strategy for addressing this goal, we have been investigating the use of hand-drawn route maps, in which the user sketches an approximate representation of the environment and then sketches the desired robot trajectory with respect to that environment. The objective in the sketch interface is to extract spatial information about the map and a qualitative path through the landmarks drawn on the sketch. This information is used to build a task representation for the robot, which operates as a semi-autonomous vehicle. Note that the task representation is based on sensing and *relative* position, not *absolute* position.

Some limited work has been done in using sketches to direct movement. Igarashi et al. proposed a path drawing technique overlaid on a virtual scene, as a means of specifying a route through the virtual environment [1]. Ferguson *et al.* developed a sketch interface for military course-of-action diagrams, which are used for strategic planning [2]. Freksa *et al.* proposed the use of a schematic map for directing robot navigation [3]. A schematic map is described as an abstraction between a sketch map and a topological map; an example is a subway map. The closest work is that of Kawamura et al. in which the user specifies a robot path by selecting via points on a sketch of the environment [4]. Artificial landmarks are placed in the scene and on the sketch to act as landmarks in the navigation process.

In this paper, we present a strategy for extracting qualitative route information from a sketched route map and then show how this information can be used for robot navigation along the sketched path. In Section II, we provide background information on human navigation and the framework for Human Robot Interaction (HRI). In Section III we present the methodology used for our approach. In Section IV we show some experiments using a Nomad200 robot simulator and we conclude with remarks on this work in Section V.

II. BACKGROUND

A sketched route map is drawn to help someone navigate along a path for the purpose of reaching a goal. An example is shown in Fig. 1a. Although route maps do not generally contain complete map information about a region, they do provide relevant information for the navigation task. People sketch route maps to include landmarks at key points along the path and use spatial relationships to help depict the route, often adding arrows and other notation for clarity [5]. The depiction of the environment structure is not necessarily accurate and may even distort the actual configuration [6]. Research by Michon and Denis [7] provides insights into how landmarks are used for human navigation and what are considered to be key route points.



(a) (b)

Fig. 1. (a) A map sketched on paper, describing a route through the MU campus. (b) A view of the actual map. Buildings 13, 9, 29 and 81 correspond to major landmarks included on the sketched map.

Skubic *et al.* describe how to extract navigation states from a hand-drawn map [8]. For the purpose of robot navigation along a path sketched on a hand-drawn map, we segment the route into qualitative landmark states (QS). QS are formed by the spatial relationships of environment landmarks with respect to the robot. Inaccuracies in the sketched maps will be corrected through motion constraints provided by the environment and executed by reactive behaviors, similar to the way in which humans use sketched maps for route navigation.

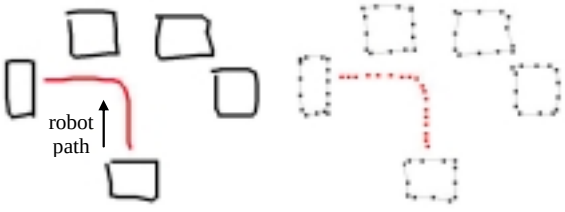
For modeling the spatial relationships between the robot and environment landmarks, we use the Histogram of Forces [8], [9], [10].

III. METHODOLOGY

A. Interpreting a PDA-Sketched Map

In this section we illustrate how navigation information is extracted from a map sketched on a PDA such as a PalmPilot. The stylus interface of the PDA allows the user to sketch a map much as he would on paper for a human colleague. The PDA captures the string of (x,y) coordinates sketched on the screen. The coordinates are monitored at a constant time interval to capture the temporal character of the sketch.

The user first draws a representation of the environment by sketching the approximate boundary of each object. During the sketching process, a delimiter is included to separate the string of coordinates for each object in the environment. After all of the environment objects have been drawn, another delimiter is included to indicate the start of the robot trajectory, and the user sketches the desired path of the robot, relative to the sketched environment. An example of a sketch is shown in Fig. 2a, and Fig. 2b shows the corresponding digital representation, where each point represents a captured screen pixel.



(a) (b)

Fig. 2. Sketch 1 (a) A route map sketched on a PDA. (b) The corresponding digital representation

B. Extracting Spatial States

The extraction of spatial states from the sketch is summarized in Fig. 3. For each point along the trajectory, a view of the environment is built, using the radius of the sensor range. For each object within the sensory radius, a polygonal region is built using the boundary coordinates of the object as vertices. To provide unique labels, landmark objects are assigned sequential numbers as they are encountered along the sketched path.

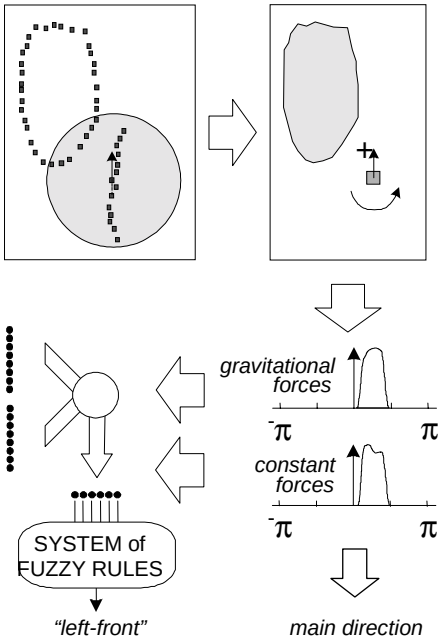


Fig. 3. Synoptic diagram showing how spatial information is extracted from the sketch

In previous work [13], we have used different strategies for building the polygonal representations of the objects, for example, using only the points of the object that fall within the sensory radius. Here, if any of the object points lies within the sensory radius, we use the entire object boundary.

Once the polygonal region of an object is built, the histograms of constant forces and gravitational forces are computed as described in [8], [9], [10]. The referent is always the robot, which is modeled as a bounding box for the histogram computations. To capture robot-centered spatial relationships, the robot orientation must also be considered. The robot's heading is computed using adjacent points along the sketched path to determine an instantaneous orientation. We compensate for the discrete pixels by averaging 5 adjacent points (centered on the considered trajectory point), thereby filtering small perturbations and computing a smooth transition as the orientation changes. After the heading is calculated, it is used to shift the histograms along the horizontal axis to produce an egocentric (robot) view.

In addition, a "main direction" is extracted from the histograms as the direction with the highest degree of truth that the object lies in this direction. The main direction is then discretized into one of 16 possible directions, as shown

in Fig. 4. The discretized main direction is used as a symbolic representation of the landmark's position with respect to the robot. Examples of some corresponding linguistic descriptions are also shown in Fig. 4.

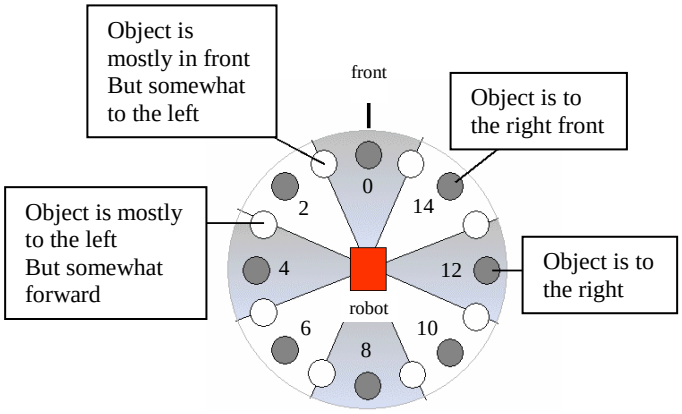


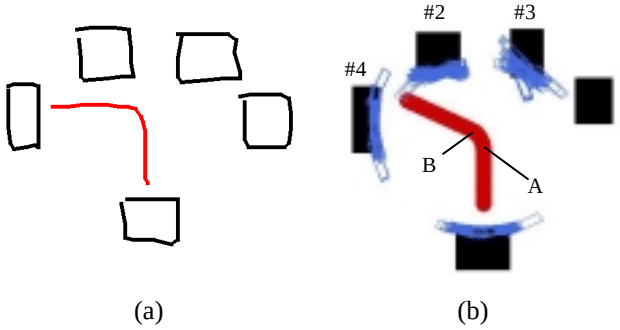
Fig. 4. Sixteen directions are situated around the robot (the small circles). The main direction of each object is discretized into one of these 16 directions (numbers 0-15).

C. Extracting Robot Movement

In addition to extracting spatial information on the environment landmarks, we also extract the movement of the robot along the sketched path. The computation of the robot heading, described above, provides an instantaneous orientation. However, we also want to track the change in orientation over time and compute what would correspond to robot commands, i.e., move forward, turn left, or turn right. The turning rate is determined by computing the change in instantaneous heading between two adjacent route points and dividing by the distance between the points to normalize the rate. A positive rate means a turn to the left, and a negative rate means a turn to the right. The robot turn commands are generated from the turning rate using a threshold (in this case, 0.7 normalized units per time step), and then filtering out spurious changes to smooth the commands. In Fig. 5a we present an example of a sketched route map on a PalmPilot PDA. Fig 7 shows the route commands extracted in the graph of the path turns. The first command is always move forward by default and the last one is always stop. In the case of Fig. 5, the intermediate commands extracted are "turn left" at trajectory step 7 and "move forward" at trajectory step 13, as shown in Fig. 6. Trajectory steps 7 and 13 of Fig. 6 correspond to points A and B of Fig. 5b respectively.

D. Extracting Landmark States at Critical Path Nodes

The discrete robot commands in Fig. 6 show the general trend in the robot movement along the sketched route and the correlation with the relative landmark positions. At the beginning of the route, when object #1 is behind the robot, the robot's movement is straight ahead. When objects #2 and #3 are in view, the robot turns to the left until #2 is to



(a) (b)

1. WHEN (An object is at direction 1 OR
   An object is at direction 2) AND
   (An object is at direction 14 OR
   An object is at direction 13)
   THEN Turn Left
2. WHEN (An object is at direction 11 OR
   An object is at direction 11) AND
   (An object is at direction 1)
   THEN Go Forward

(c)

Fig. 5. (a) PDA Sketch 1. (b) Route executed by the robot on the simulator. (c) Sequential compilation of steps after extracting significant landmark states from critical path nodes.

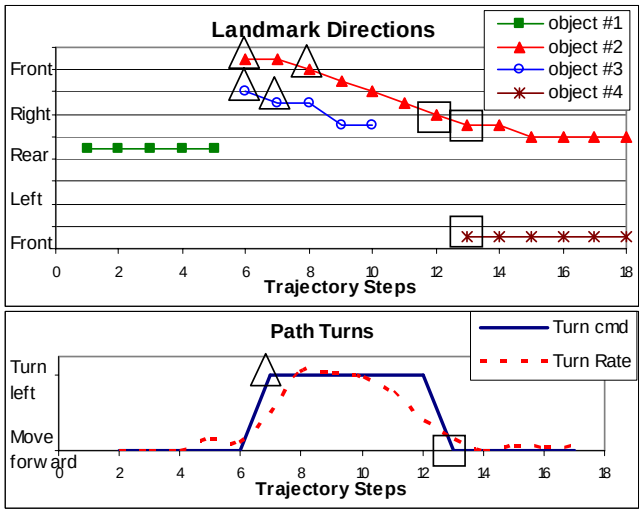


Fig. 6. PDA Sketch 1. (Top) Discrete main directions of the objects in view for every trajectory step of the sketched route. (Bottom) Normalized turning rate and turn commands of the robot. Critical path nodes and associated landmark states are identified with open square and triangle symbols.

the right and object #4 appears in front of the robot. The robot begins to move forward again and it stops when it gets close to #4.

The starting and ending states and the states at which the robot changes its turning rate comprise the critical path nodes. Although this seems straightforward for people to interpret, it is difficult to extract the essential information in a robust way and invariant to shape and scale, especially since we want the system to produce what people would consider as important. To incorporate knowledge of human navigation and the qualitative nature of the spatial information, a system of fuzzy rules is used to extract the significant landmark states at each critical node. In fact, what is needed is the change in landmark state, *i.e.,* the landmark event. Three linguistic variables are used as inputs: (1) the event timing, which is a measure of the closeness of the landmark event to the critical path node (measured in time steps), (2) the main direction of the landmark (0-16), and (3) the object status, *i.e.*, the object has just come in view, the object has disappeared, or no change in status. The output variable is the event match, a confidence measure of how significantly this landmark event matches the critical path node. The events with a high level of confidence (i.e. above a pre-specified threshold) are used to identify the landmark state associated with the critical path node.

Fig. 7 shows the rules used and Fig. 8 shows the membership functions. Landmarks in the front have a higher match than those on the sides or in the rear. Landmarks that have just come in view are given a higher match, as are events that are closer in time to the critical node. The results for Sketch 1 (Fig. 5a) are shown in Fig. 5c. If more than one landmark events are identified as significant for a particular critical node, they all get combined into a single statement using the OR operator for the same object and the AND operator for different objects. The qualitative instructions for the robot are given in the form of:

When <landmark state> *Then* <robot command>

If <*event Timing*> is <*very-close*>
then <*event match*> is <*very-high*>
If <*event Timing*> is <*close*>
then <*event match*> is <*good*>
If <*event Timing*> is <*far*>
then <*event match*> is <*poor*>

If <*main Direction*> is <*front*>
then <*event match*> is <*very-high*>
If <*main Direction*> is <*left*>
then <*event match*> is <*high*>
If <*main Direction*> is <*right*>
then <*event match*> is <*high*>
If <*main Direction*> is <*rear*>
then <*event match*> is <*poor*>

If <*object Status*> is <*new*>
then <*event match*> is <*very-high*>
If <*object Status*> is <*disappeared*>
then <*event match*> is <*good*>

Fig. 7. Rules for extracting critical path nodes.

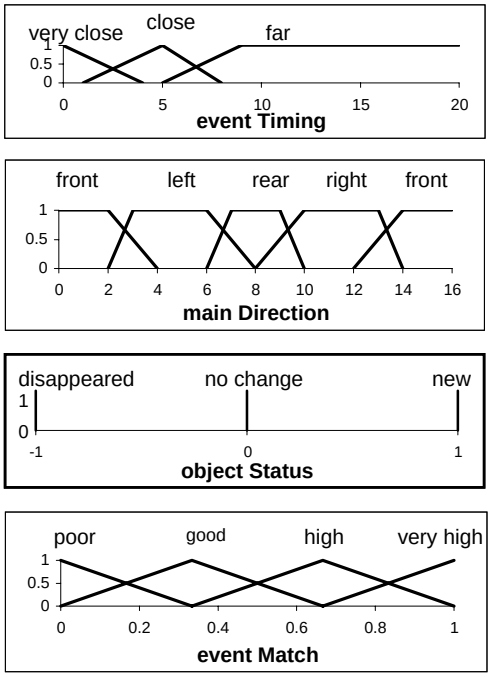


Fig. 8. Membership functions for extracting qualitative states at critical path nodes

The path description of Sketch 1 is then a sequential compilation of the steps, as shown in Fig. 5c.

E. Traversing paths with a robot

The sequential compilation of steps, mentioned above, comprise the instructions that the robot has to follow to reach its goal. All instructions are based on landmark states. We used a Nomad200 robot simulator to execute the sequential steps in linguistic terms in an attempt to verify that the robot will indeed perform the task it was intended to. On the Nomad200 simulator we solely use the 16 sonar sensor array for landmark state recognition. Note that recognition of particular objects (i.e. “Object #3”, or “Round Table”) is not possible by using only the sonar array. Consequently, when a particular landmark state is matched, the corresponding navigation command will be executed,

even though the matched state may be only similar to the one sketched on the PDA.

For safety purposes the robot is also running a low-level obstacle avoidance algorithm. The output of the obstacle avoidance algorithm is fused with the output of the high level instruction following algorithm. Obstacle avoidance is based on potential fields and achieved by the following formula:

$$S = \frac{w(s) \cdot C \cdot \sin(s \cdot \frac{p}{8})}{e^{proximity(s)}}$$

Where:

S is the measure of the potential fields direction vector

w(s) is the weight of each sonar sensor s

C is a constant that puts S into a suitable for our purposes range

proximity (s) is the distance returned by sensor s in tenths of an inch

The 7 sonar sensors that point to the rear of the robot carry a weight of 0, so that objects behind the robot have no influence on the robot's navigation path. The rest of the sensors carry weights between 0.4 and 1.0 spread out symmetrically between the sensors, with the forward sensor having a weight of 1.0. This way we ensure that objects that obstruct the robot's path the most, carry the most weight.

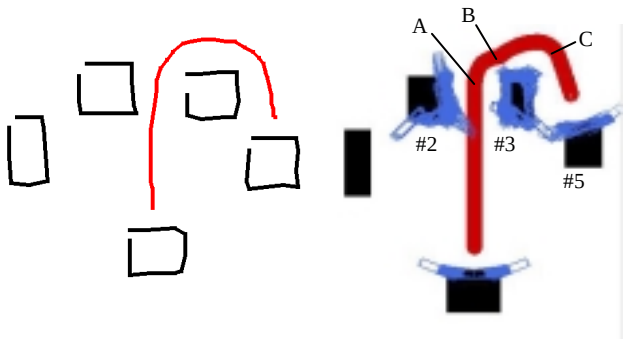
In order to execute the instructions generated by analyzing the sketch, the robot begins to move forward and checks the environment through the sonar sensors to match the first landmark state. When such a state is matched, the corresponding command is executed and continues to be executed until the robot matches the next landmark state. In the example of Fig. 5, when the robot matches state 1, it will turn left until state 2 is matched. At that point, it will start going forward. A "stop" command is issued when all landmark states have been matched and the robot is at a pre-specified (close) distance from any object in front of it.

In the case of Fig. 5, the conditions for an object at direction 1 and an object at direction 14 got matched at point A of the robot's route, by objects #2 and #1 respectively. At point A the robot starts making a left turn until the conditions of state 2 are matched. Objects #2 and #3 match the conditions for directions 1 and 11 respectively and the robot executes the "GO FORWARD" command. The stop condition is matched when the robot gets close enough to object #4.

IV. EXPERIMENTS AND RESULTS

A different sketch on the PDA is shown in Fig. 9a. Fig. 9b shows the execution by the robot on the simulator, and Fig. 9c shows the steps extracted by the fuzzy rule base.

At point A of the robot's route, objects #2 and #3 are at directions 5 and 11 respectively. This landmark state is identified as state 1, and the corresponding command to turn



(a) (b)

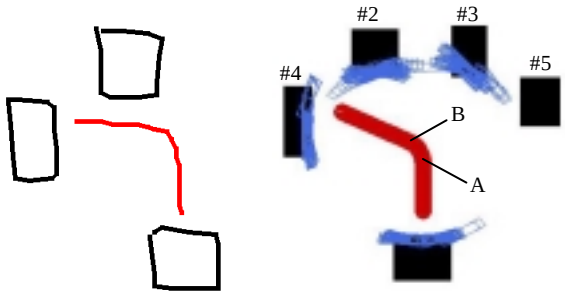
1. WHEN (An object is at direction 3 OR
   An object is at direction 5) AND
   (An object is at direction 6 OR
   An object is at direction 11)
   THEN Turn Right
2. WHEN (An object is at direction 1 OR
   An object is at direction 0)
   THEN Go Forward

(c)

Fig. 9. (a) PDA Sketch 2. (b) Route executed by the robot on the simulator. (c) Sequential compilation of steps after extracting significant landmark states from critical path nodes.

right is executed. As the robot turns to the right, the turn is a bit too sharp to let it go around object #3. When it starts heading towards object #3 and it gets too close to it, the obstacle avoidance behavior makes it turn to the left at point B. As it turns away from object #3 and moves further away, the higher level navigation command of turning to the right carries more weight than the obstacle avoidance command and makes the robot start turning to the right again, to go around object #3. At point C, object #5 is detected at direction 0, which makes the robot match landmark state 2 and start to move forward. When the robot gets close enough to object #5, it stops.

To demonstrate that additional objects in the environment do not alter the robustness of the algorithm, we created the sketch of Fig. 10a. We then made the robot run in the environment of Fig. 10b (which is identical to the one in Fig. 5 and Fig. 9). The robot executed the route successfully, without being influenced by objects #3 and #5. The extracted landmark states in this case are shown in Fig. 10c.



(a) (b)

1. WHEN An object is at direction 2
   THEN Turn Left
2. WHEN (An object is at direction 11 OR
   An object is at direction 11) AND
   (An object is at direction 1)
   THEN Go Forward

(c)

Fig. 10. (a) PDA Sketch 4. (b) Route executed by the robot on the simulator. (c) Sequential compilation of steps after extracting significant landmark states from critical path nodes.

V. CONCLUDING REMARKS

We have presented a way to navigate a robot using PDA sketches. We included a basic demonstration of our approach using a Nomad200 robot simulator. This work is part of an ongoing research on sketch-based robot navigation.

Other areas of this research that need to be addressed include object recognition for a more accurate identification of landmark states and a more robust algorithm for navigating between landmark states. An algorithm based on triangulation of the robot's position based on landmarks such as the one described in [4] may yield more efficient navigation between landmark states.

REFERENCES

[1] T. Igarashi, R. Kadobayashi, K. Mase, H. Tanaka, "Path Drawing for 3D Walkthrough", in *Proceedings of the 11th Annual Symposium on User Interface Software and Technology*, ACM UIST 1998, San Francisco, CA, Nov., 1998, pp. 173-174.

[2] R. Ferguson, R. Rasch, W. Turmel, and K. Forbus, "Qualitative Spatial Interpretation of Course-of-Action Diagrams", in *Proceedings of the 14th International Workshop on Qualitative Reasoning*, Morelia, Mexico, 2000.

[3] C. Freksa, R. Moratz, and T. Barkowsky, "Schematic Maps for Robot Navigation", in *Spatial Cognition II: Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, C. Freksa, W. Brauer, C. Habel, K. Wender (ed.), Berlin: Springer, 2000, pp. 100-114.

[4] K. Kawamura, A.B. Koku, D.M. Wilkes, R.A. Peters II and A. Sekmen, "Toward Perception-Based Navigation Using Egospheres", *International Journal of Robotics and Automation*, to appear.

[5] B. Tversky and P. Lee, "How Space Structures Language," in *Spatial Cognition: An Interdisciplinary Approach to Representing and Processing Spatial Knowledge*, C. Freksa, C. Habel, K. Wender (ed.), Berlin: Springer, 1998, pp. 157-176.

[6] B. Tversky, "What do Sketches Say about Thinking?" 2002 AAAI Spring Symposium, Sketch Understanding Workshop, Stanford University, AAAI Technical Report SS-02-08, March, 2002, pp. 148-151.

[7] P.-E. Michon and M. Denis, "When and Why Are Visual Landmarks Used in Giving Directions?" in *Spatial Information Theory: Foundations of Geographic Information Science*, D. Montello (ed.), Berlin: Springer, 2001, pp. 292-305.

[8] M. Skubic, P. Matsakis, B. Forrester and G. Chronis, "Extracting Navigation States from a Hand-Drawn Map," in *Proceedings of the IEEE 2001 International Conference on Robotics and Automation*, Seoul, Korea, May, 2001, pp. 259-264.

[9] P. Matsakis, J. Keller, L. Wendling, J. Marjamaa, and O. Sjahputera, "Linguistic Description of Relative Positions in Images", *IEEE Trans. on Systems, Man and Cybernetics*, part B, vol. 31, no. 4, pp. 573-588, 2001.

[10] M. Skubic, P. Matsakis, G. Chronis and J. Keller, "Generating Multi-Level Linguistic Spatial Descriptions from Range Sensor Readings Using the Histogram of Forces," *Autonomous Robots*, to appear January 2003.