Behaviour-Based Docking using the DAMN Arbiter

T. Bailey, E. M. Nebot, J. K. Rosenblatt and H. F. Durrant-Whyte

Department of Mechanical and Mechatronic Engineering

The University of Sydney, NSW. 2006

e-mail: tbailey/nebot/julio/hugh @mech.eng.usyd.edu.au

Abstract

This paper discusses a method that enables reliable docking of a mobile robot with a rectangular container. The process involves an aligned approach to the container while avoiding any obstacles in the region and avoiding collision with the container itself.

A modular behaviour-based architecture called the Distributed Architecture for Mobile Navigation (DAMN) is used. Raw sensor data is processed to produce robust virtual sensors that provide input data to the behaviour modules. Centralised arbiters then asynchronously process the behaviour outputs and determine the set points for the mobile robot drive and steering actuators.

Testing of the virtual sensors and behaviour-based algorithms was performed on an indoor mobile robot, SydNav, with wheel-encoders and a scanning range laser as its sensors. Further testing of a container pose-estimation virtual sensor took place on a quayside cargo-handling vehicle (a straddle-carrier); again using scanning lasers for its sensorial input.

1 Introduction

Automatic pickup or manipulation of an object has been considered previously in terms of a kinematic arm and end-effector on a mobile robot base [Mandel and Duffie, 1987][Nagatani and Yuta, 1996]. In this case the alignment of the mobile robot is non-critical as the arm mechanism has sufficient degrees of freedom to accomplish the task irrespective of the vehicle's pose. The concept of autonomous pickup via an aligned approach and docking of a mobile robot, with its subsequent kinematic constraints, requires a more stringent approach method.

An aligned approach does not lend itself easily to a deliberative solution in which the container has known global coordinates and the vehicle calculates an optimal trajectory to achieve the docking procedure. Errors in the initial container pose estimate and in the vehicle's global pose estimate make this a fragile method at best. However, with closed-loop sensor information, reasonable robustness can be achieved. Sensors detect the relative pose of the container with respect to the vehicle and update the container pose estimate so that the trajectory plan can be adjusted accordingly. The tests carried out on the straddle-carrier vehicle used this control method.

A more reactive behaviour-based control method is better suited to the docking problem. No trajectory plan is made in this case but control actions are based on the outputs of a set of behaviour modules. A wide range of behaviour-based systems have been designed to date from purely reactive, notably Brook's subsumption architecture [Brooks, 1986], to hybrid combinations of traditional deliberative planners and reactive low-level controllers, such as Arkin's AuRA (Autonomous Robot Architecture) [Arkin, 1997]. In this paper, a version of hybrid architecture called the Distributed Architecture for Mobile Navigation (DAMN) [Rosenblatt, 1997] is used. DAMN consists of a distributed system of behaviour modules which are interpreted in an intelligent manner by a centralised arbitration module.

Often it is undesirable for all behaviours to be active at once. Methods such as RAPs (Reactive Action Packages) [Firby, 1996] and Hughes' behaviour pool [Payton, 1986] enable a selection of behaviours to be activated for a particular task while the remaining behaviours are dormant. The DAMN architecture uses a mode manager which is effectively a state machine that selects a number of behaviours for the task at hand in a similar method to Hughes' behaviour pool. The mode manager can also apply dynamic weightings to each behaviour.

The purely reactive approach to processing raw sensor data is to have such a high bandwidth update rate that erroneous data can be effectively ignored. In this paper, however, raw sensor data is fused and filtered to provide modules of higher-level information so that later processes using this information can function reliably at much lower bandwidths. These modules of processed sensor data have been termed *virtual sensors*. The advantages are twofold. First the output of the virtual sensor is more robust as spurious values can be rejected and estimates modeled and filtered. Secondly, the output of the virtual sensor is consistent irrespective of the type of input sensor providing it with information.

This paper is organised as follows. Part 2 describes the vehicles from which experimental data was collected to test the virtual sensors and behaviour/arbiter algorithms. Part 3 describes the virtual sensor modules in terms of the raw sensor data inputs, signal processing algorithms, and optimised outputs. The next section briefly describes the behaviour modules that were tested for the docking process and section 5 describes the DAMN arbiters that choose the best steering angle and velocity set points based on the behaviour data. The final section makes concluding remarks on the operation of the system elements.

2 Test Vehicles

The majority of testing was performed on an indoor mobile robot, SydNav. All behaviour-based algorithms were tested on this vehicle. Some additional testing for a version of the Container Pose virtual sensor was done on an outdoor cargo-handling vehicle – a quayside straddle-carrier.

2.1 Indoor Mobile Robot - SydNav

SydNav (see Figure 1) is a three-wheeled robot with the two rear wheels fixed in the forward direction and able to rotate freely. The front wheel only provides both drive and steering (a tricycle model). This arrangement is relatively free from slip and produces quite accurate odometric estimates.

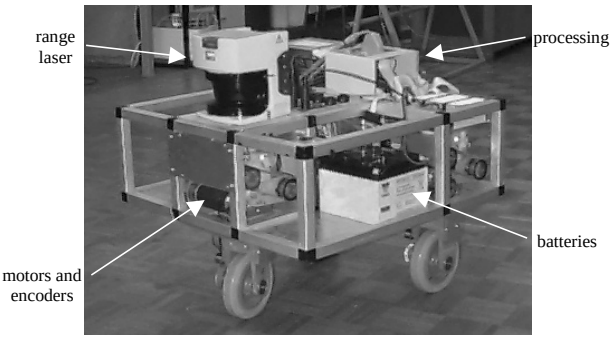

Figure 1: SydNav Vehicle

Sensor data is supplied by wheel encoders, mounted on the shafts of the steering and drive motors, and a SICK 2D scanning range laser mounted on the front of the vehicle.

The wheel encoders have resolution of 2000 counts per revolution which can be transformed directly into steering angle (γ) and distance traveled (Δd) by considering the drive train gear ratios and the wheel radius.

The scanning laser produces a 2D scan over 180°, with resolution 0.5°, for a range of about 30 meters. Its short 900nm wavelength means that it obtains reliable diffuse reflection off nearly all surfaces. There are, however, quantization errors of ±30 mm; false readings (artifacts) along steep edges; and possible false reflections from rain, steam or dust particles. These must be removed by filtering.

2.2 Outdoor Mobile Robot - Straddle Carrier

In a port environment, quay cranes move containers from the ship to the dock. A straddle carrier (see Figure 2) is a 13m tall cargo-handling vehicle used to transfer containers from beneath the crane to a storage stack. It operates by driving over (straddling) the container and dropping a spreader onto it. The kinematics of this vehicle are characterized by its anti-symmetric steering meaning that for any fixed steering angle the front and rear wheels travel along the same arc.

The current straddle-carrier test vehicle already has a navigation system in place that gives it a global pose estimate. This means that reliable global pose is an available resource to any of the modules tested on this vehicle. It also has two 2D scanning range lasers mounted on the front left-hand-side and rear right-hand-side of the vehicle. These lasers are similar to the one on SydNav but have a sweep of just 100° and resolution of 1°. They are angled inwards by 25° to enable a view of the container at close range.

Figure 2: Straddle Carrier Vehicles

3 Virtual Sensors

Sensor data for the behaviour modules is provided by a collection of virtual sensors. These are derived from raw sensor data that has been filtered and fused to produce a robust data set. The outputs of the virtual sensors are tailored to the specific input requirements of the behaviour modules and are designed to produce a known output type regardless of the particular sensor types providing them with raw information.

There are three virtual sensors discussed in this paper. The first two, Global Position and Obstacle Map, were tested on SydNav only. The third, Container Pose, was tested on both SydNav and the straddle-carrier vehicle.

3.2 Global Position Virtual Sensor

The Global Position Virtual Sensor is essentially a Kalman filtered estimation of the vehicle's pose (x, y, ϕ) fusing wheel-encoder data with laser observations (see Figure 3). Static features, small clusters and lines, that appear stable over several observations are used as external reference points to correct pose estimate errors. An Extended Kalman Filter (EKF) is used to fuse external observations with odometric data. The state and observation models used for the filter are based on the equations in [Borenstein, 1996]. Presently, the environment is arranged to make this process as simple and reliable as possible by placing several isolated objects within the observable region to serve as beacons.

The Global Position virtual sensor is used to maintain a global representation for the steering arbiter (Section 5.2).

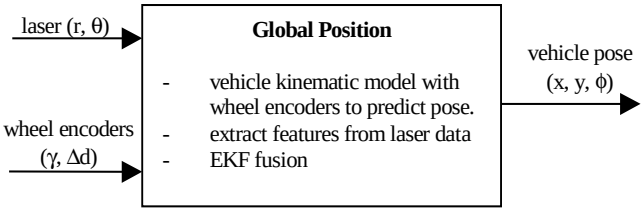

Figure 3: Global Position Virtual Sensor

3.3 Obstacle Map Virtual Sensor

The obstacle map is a local map, in vehicle-centered coordinates, that encompasses a rectangular region surrounding the vehicle. The map extents are determined by such parameters as: the size of the operating environment; the velocity of the vehicle; and other physical characteristics of the vehicle. For example, the straddle-carrier operates in both forward or reverse so the map should be symmetrical in both front and rear.

Scanned data from the laser is stored in a list if the points lie within the map range. For each update of the vehicle pose, the stored data points are transformed according to the relative change in pose, enabling non-observed objects to be remembered. New scans are added to the list as they arrive. Any data points that are no longer within the bounded region are deleted from the list (see Figure 4).

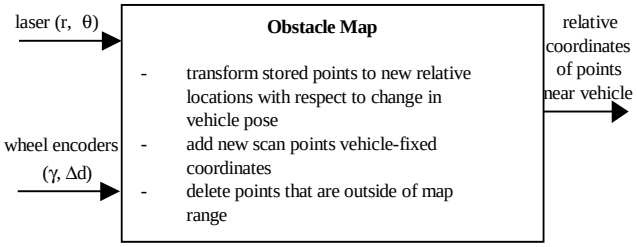

Figure 4: Obstacle Map Virtual Sensor

The implementation of this virtual sensor on SydNav is shown in Figure 5. The area in front of the vehicle is updated with each new 180° scan while the area behind the vehicle is filled with points seen previously and moved backwards as the vehicle travels forward. As with the Global Position virtual sensor, this is a very simplistic virtual sensor and relies on good odometry and fairly noise-free laser readings. Nevertheless, it is sufficient to demonstrate the DAMN-based docking procedure and show, at least conceptually, the encapsulation of raw sensor data into a virtual sensor module.

3.4 Container Pose Sensor

The Container Pose Virtual Sensor operates as follows. An approximate estimate of the container pose is given once the vehicle is within the region where the container might be observable. Each new scan the data points are divided into clusters. Those clusters that lie close to the predicted location of the container have lines fitted to them using a least-squares algorithm. The least-squares algorithm used here is a Cartesian-based algorithm similar to that found in [Kahn et al., 1990].

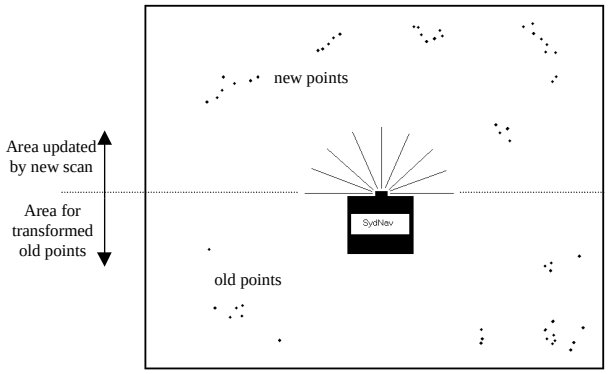

Figure 5: Obstacle Map Implementation

Geometric constraints of length and slope are used to verify the observation of the front corners of the container, which is then used to update an EKF estimate of the relative container pose (X, Y, Φ) maintained at the container centre (see Figure 6).

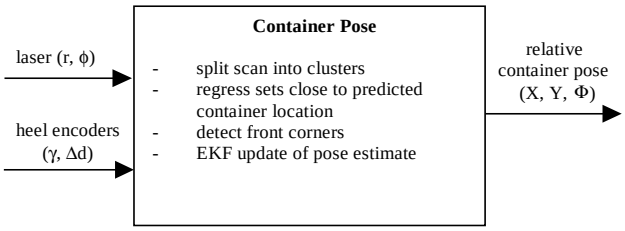

Figure 6: Container Pose Virtual Sensor

Figure 7 shows part of an actual scan from the laser in an outdoor test environment. The environment consisted of a single cargo container in the presence of quite a lot of natural clutter. In this scan test, linear regression was performed over the whole data set, without constraints for expected pose, in order to test the robustness of the container recognition algorithm. The figure shows the container directly in front of the laser with miscellaneous objects to the right-hand-side. In the tests carried out at this site, the container was consistently detected while moving the laser through the environment.

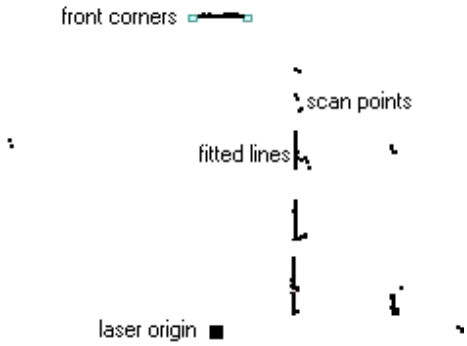

Figure 7: Linear Regression to Actual Laser Scan

The state model for the Container Pose virtual sensor assumes the container is stationary and the change in relative pose is due to the vehicle motion:

$$\hat{\mathbf{x}}(k|k-1) = \begin{bmatrix} X(k|k-1) \\ Y(k|k-1) \\ \Phi(k|k-1) \end{bmatrix} = \begin{bmatrix} X(k-1|k-1) - \Delta d\cos(\gamma) \\ Y(k-1|k-1) - \Delta d\sin(\gamma) \\ \Phi(k-1|k-1) - \dfrac{\Delta d\sin(\gamma)}{B} \end{bmatrix} \quad (1)$$

Here, the state is the pose of the container and the change in state is caused by the movement of the vehicle (Δd) and its steering angle (γ). The rate of change in (Φ), as defined in Equation 1, is based on a tricycle model vehicle where (B) is the wheelbase.

Figure 8 shows the observation model for the Container Pose virtual sensor. The observation vector consists of the range (r) from the laser to a perpendicular intersection with the front edge of the container and the angle (θ) of this range vector. It also incorporates the distance (d1 and d2) from the intersection to the front corners, given the fixed width (2W) and length (2L) of the container.

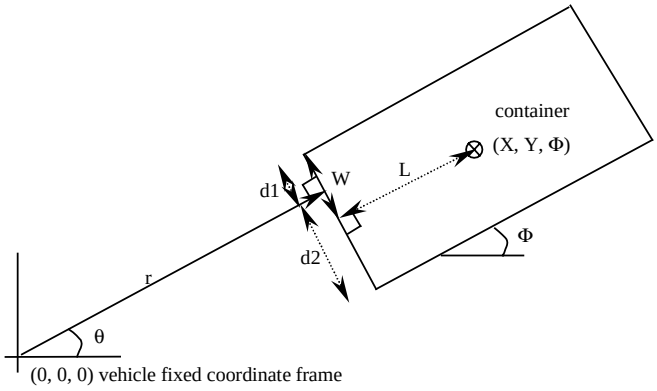

Figure 8: Observation Model

An observation of the front corners can be inferred without actually seeing the entire front. A partial view of the front and one side is sufficient to calculate the position of the corners.

The observation model for the container allows for the uncertainty of the front corners to be incorporated as follows:

$$\mathbf{z}(k) = \begin{bmatrix} r \\ \theta \\ d1 \\ d2 \end{bmatrix} = \begin{bmatrix} X\cos(\Phi) + Y\sin(\Phi) - L \\ \Phi \\ W + Y\cos(\Phi) - X\sin(\Phi) \\ W - Y\cos(\Phi) + X\sin(\Phi) \end{bmatrix} \quad (2)$$

The noise injected into the observation of r and θ is directly proportional to the number of points contained in the linear regression. The noise injected for d1 or d2 is based on what part of the container was actually observed. If the first corner is seen directly and the second is inferred then the noise for d1 is small and d2 is set large as its observation is correlated to the first corner.

4 Behaviour Modules

There are two sets of behaviour modules described here. The first is for the control of steering angle (γ) and the second is for control of velocity (V). The behaviour output type is different for these two sets and so they are processed by two separate and independent arbiters.

4.1 Steering Behaviours

Steering behaviours produce outputs of lines, points or polygons within the global 2D Cartesian space of the robot's environment. These lines, points or polygons are assigned a positive or negative utility where positive utility indicates a favourable state and negative utility indicates an unfavourable state. A utility is simply a numerical value (between ±∞) that describes the degree of favourability.

Head To Goal A very simple behaviour where a goal point (x, y) is specified and the behaviour applies a fixed positive utility to it.

Avoid Obstacles Points from the Obstacle Map virtual sensor are transformed to global coordinates using the global pose estimate. They are clustered together and rectangles are drawn around each cluster (see Figure 9). The rectangles are each assigned a large negative utility.

Dock The relative container position estimate from the Container Pose virtual sensor is used to calculate a line of positive utility passing through the middle of the container and parallel to its sides. This will tend to align the vehicle with the container as it approaches.

4.2 Velocity Behaviours

Velocity behaviours operate in 1D space whereby they specify the maximum allowable velocity of the vehicle in both forward and reverse (negative) directions.

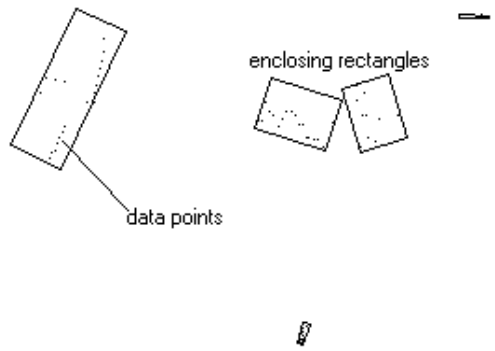

Figure 9: Enclosed Obstacle Map Point Clusters

Head to Goal The velocity bound in the direction towards the goal is given some nominal value. The velocity bound in the direction away from the goal is small but non-zero to allow for possible maneuvering due to obstacles. As the distance to the goal approaches zero, the velocity bounds shrink to zero causing the vehicle to stop.

Avoid obstacles This behaviour has nominal velocity bounds in both directions. The velocity bound in the direction towards an obstacle shrinks according to the inverse distance squared to the obstacle.

Dock The velocity bound in the forward direction is set to some nominal value and the bound in the reverse direction to a small non-zero value. Information from the Container Pose virtual sensor determines whether the docking process becomes impossible (i.e. insufficient alignment) and the forward velocity bound shrinks towards zero and the reverse velocity bound is increased. Once sufficient alignment has been attained, the original velocity bounds are reinstated.

5 DAMN Arbiter

The general structure of the DAMN arbitration system is shown in Figure 10. The arbiter has inputs from the behaviour modules and the behaviour mode manager. The output from the arbiter is a set point value for a particular controller based on the information from all the active behaviour modules.

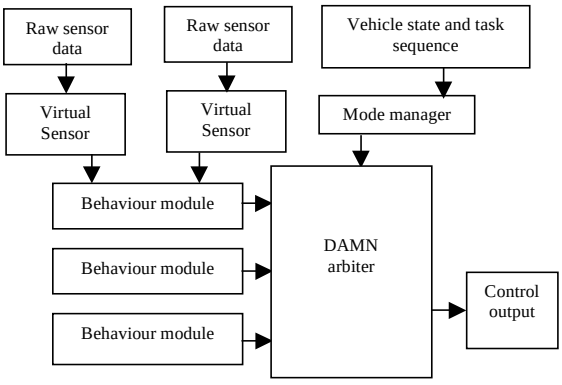

Figure 10: System Block Diagram

5.1 Behaviour Mode Manager

The Mode Manager is a state machine that can dynamically assign weightings to each of the behaviour inputs. Presently, the Mode Manager simply switches behaviours on or off (which is the same as applying weightings of one or zero respectively). The Mode Manager states for the current SydNav configuration are shown in Figure 11.

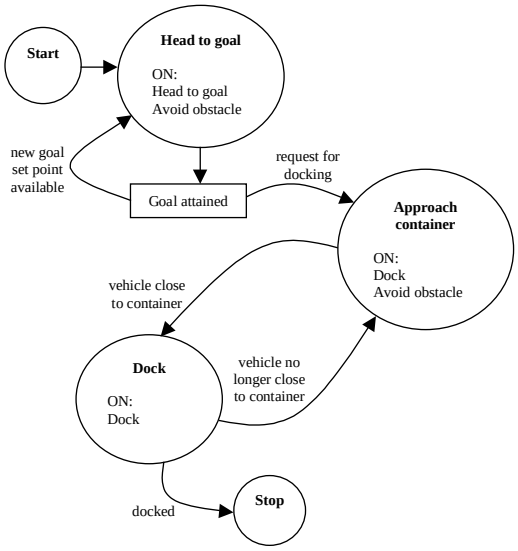

Figure 11: Mode Manager States

5.2 Steering Arbiter

The steering arbiter is a utility arbiter [Rosenblatt, 1998] which operates in the following manner. Taking the vehicle's current pose in global coordinate space, the pose of the vehicle at a fixed time-step in the future is predicted for a variety of steering angles. The steering set is limited by the kinematic constraints of the vehicle from hard-left to hard-right (see Figure 12) and an arbitrary resolution is given between these two limits.

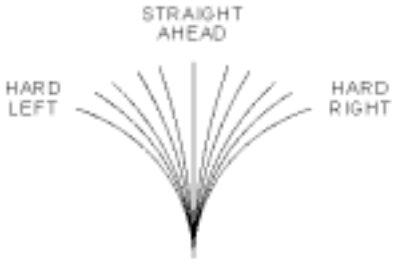

Figure 12: Range of Steering Angles taken from [Rosenblatt, 1997]

For each predicted pose, the probability of hitting each utility point, line or polygon is calculated. The most desirable steering angle is then calculated using the following formula [Berger, 1985]:

$$(a) = \sum_{c} U(c) P(c|a,e) \qquad (3)$$

U(a) stands for the utility of a particular action (a) where the action, in this case, is a steering angle (γ_i). U(c) is the utility of a particular consequence (c) where the consequence, in this case, is hitting a certain point, line or polygon. Both U(c) and the location of the points, lines, or polygons are defined by the behaviour modules. The notation P(c|a,e) means the probability of a consequence (c) given the action (a) and the evidence (e). The evidence (e), in this case, is the location of the point, line or polygon that makes possible the consequence (c).

Thus, the steering angle (γ_i) gives a predicted pose that has the probability P(c|a,e) of a particular consequence (c). The probability of each consequence multiplied by its utility is summed for each possible consequence to give the utility of the action U(a). The action with the highest utility is chosen as a steering set point.

The difficult part of this algorithm is determining a valid equation for calculation of P(c|a,e). Previously in the DAMN architecture this was a function of distance from the evidence (e). For a docking procedure it becomes necessary to extend this to being a function of distance, orientation and velocity. A good function in this vein is the General Potential Fields obstacle avoidance equation [Krogh, 1984]:

$$GPF = \frac{\alpha v}{2d\alpha - v^2} \tag{4}$$

Here α stands for the acceleration limit of the vehicle; d is the distance from the vehicle to the closest point of the obstacle; and v is the magnitude of the vehicle's velocity component in the direction towards the closest point of the obstacle. If v is less than zero then the GPF is set to zero.

The characteristics of the GPF function seem valid for the purposes of calculating P(c|a,e) and so it is used directly in the current steering arbiter design:

$$P(c|a,e) = GPF \tag{5}$$

5.3 Velocity Arbiter

The velocity arbiter is much simpler and, hence, much higher bandwidth than the steering arbiter. It simply takes the plus-minus maximum velocity bounds from each behaviour and chooses the maximum magnitude from the intersecting set. It is possible that in some circumstances the velocity may oscillate between forward and reverse but further experimentation is necessary to determine this.

6 Conclusions

The docking process for a mobile robot is implemented based on the DAMN behaviour-based architecture. Virtual sensors minimise sensor error and produce optimal estimates in the required format. A distributed system of behaviour modules independently provides information on what actions should be taken such that data is gathered without bottlenecks. Centralised arbiters interpret the information for the behaviours so that all available data is considered in making an action decision.

The architecture described in this paper enabled robust collision-free docking with a container in the presence of other obstacles.

Further research is required to ensure the stability of the forward/reverse component of the velocity arbiter. Also an extension of the function for calculation of P(c|a,e) should be made to incorporate the uncertainties from the information gathered by the virtual sensors and behaviours which result in uncertainty of the vehicle state and object locations.

References

[Arkin, 1997] Arkin, R.C. and Balch, T., "AuRA: Principles and Practice in Review," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 9, No. 2, 1997.

[Berger, 1985] Berger J., "Statistical decision theory and Bayesian Analysis", 2nd ed. New York: Springer, 1985.

[Borenstein, 1996] Borenstein J., "Navigating Mobile Robots: Systems and Techniques", A. K. Peters, Ltd., 1996

[Brooks, 1986] Brooks R. A., "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 14-23, 1986.

[Firby, 1996] Firby R. J., "Modularity Issues in Reactive Planning", *Proceedings of the Third International Conference on AI Planning Systems*, AIPS-96, Edinburgh Scotland, pp 78-85. Brian Drabble ed., AAAI Press, May 1996

[Kahn et. al., 1990] Kahn P., Kitchen L., Riseman E. M., "A Fast Line Finder For Vision-Guided Robot Navigation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 11, pp. 1098-1102, November 1990.

[Krogh, 1984] Krogh B. H., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *SME-RI Technical Paper MS84-484*, August 1984.

[Mandel and Duffie, 1987] Mandel K., Duffie N. A., "On-Line Compensation of Mobile Robot Docking Errors", *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 6, pp. 591-598, December 1987.

[Nagatani and Yuta, 1996] Nagatani K., Yuta S., "Door-Opening Behavior of an Autonomous Mobile Manipulator by Sequence of Action Primitives", *Journal of Robotic Systems*, vol. 13, no. 11, pp. 709-721, 1996.

[Payton, 1986] Payton D. W., "An Architecture for Reflexive Autonomous Vehicle Control", *IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 1838-1845, April 1986.

[Rosenblatt, 1997] Rosenblatt J., "The Distributed Architecture for Mobile Navigation", Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2/3, pp.339-360, April-September, 1997.

[Rosenblatt, 1998] Rosenblatt J., "Utility Fusion: Map-Based Planning in a Behavior-Based System", in Field and Service Robotics, Springer-Verlag, 1998.