AMOR - An Autonomous Mobile Robot Navigation System

Hans Blaasvær, Paolo Pirjanian and Henrik I. Christensen
Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, Fr. Bajers Vej 7, Building D1, DK-9220 Aalborg, Denmark.
{habl,paolo,hic}@vision.auc.dk

1 Introduction

In a wide range of application areas ranging from domestic robotics and warehouse management to surveillance it is desirable to have autonomous mobile robot systems that can perform specific tasks such as cleaning, map generation, or delivery. Researchers have worked on the problem for more than 3 decades. This has lead to a number of successful systems, in areas such as warehouse delivery and semi-autonomous bomb removal. It is, however, characteristic that almost all application systems can only operate in a very limited application domain under severe constraints. A good impression of the state of the art in the research area can be found from a review of the annual AAAI robotics contest [4]. In the 1993 contest most robots were built specific to participate in the contest with extensive knowledge about the domain, which makes it possible to build systems which only use simple sensing modalities such as sonars. For operation in a less known environment it is judged necessary to employ sensing modalities that provide more detailed information so that unknown structures can be identified and avoided. One such sensing modality is computer vision.

To verify that it is possible to build competent mobile robotic systems, that can perform realistic tasks with a minimum of domain knowledge an autonomous system, which exploits vision as an integral part of it sensing and planning has been constructed. To enable comparison of the performance of the robot to that of other systems it was chosen to use the tasks from the AAAI contest as a basis for definition and operation of the system. In 1993 the contest contained three tasks: 'escape from the office', 'office delivery' and 'office rearrangement'. The first two tasks can be combined into a single task of office delivery. The last task requires simple sensing for detection and description of box type objects. This task requires use of a large operation area, which we unfortunately don't have available. We have consequently focussed on design and implementation of a system that should be able to perform office delivery in a natural office or laboratory environment. It is here essential that the environment is not modified in any way to accommodate the operation of the robot. In the remaining of this paper it is described how such a system may be implemented. First the system architecture is described, followed by a description of the system modules, and finally the implementation and a set of experiments are described in detail.

2 System Architecture

The office delivery system is organized as a hybrid architecture, combining reactive behaviors for local short-term planning; and supervision by a deliberative strategic executive, which runs concurrently with the reactive subsystem, and performs goal-directed, task-achieving planning. This organization imposes sufficient reactivity for handling contingencies, and thus ensures safe collision-free navigation, while supporting deliberative behaviors for task-achieving purposes [12].

Figure 1 depicts the overall architecture of AMOR divided in four subsystems: *Pilot*[1], which performs the high-level planning as well as resource scheduling and control activities; *Navigation Subsystem*, which provides a set of executable actions ranging from perception to action activities; *Reactive Subsystem*, which implements a set of reactive behaviors for safe navigation, through a tight coupling of perception and action; and *Hardware Interface*, which defines a high-level interface to the sensors and actuators of the physical part of the system.

Given a task issued by a human operator, the Task Planner synthesises a plan, a sequence of high-level subtasks, based on an a priori provided graph representation of the navigation environment, which when executed successfully will achieve the overall task. The assumption made by the Task Planner is that navigation through a door is principally independent of how the (preceding) room navigation task has been accomplished. Pursuing this line of thought, the office delivery task is decomposed in ***room navigation***, ***door navigation*** and ***hallway navigation***. Each navigation *context* imposes different requirements on the strategy of navigation regarding precision, speed, and reactivity. For instance, the demand on precision is higher when driving through a doorway than when navigating in a room. Given a subtask and the state of the world, the Context Selector selects the navigation context that defines a strategy for navigation in that context. A set of modules, in the Navigation Subsystem, corresponding to the navigation context, will subsequently be initiated. The modules run concurrently and are coordinated by the Pilot to accomplish the given subtask. The modules in the Navigation Subsystem can be parameterized to adapt to a given navigation context. The set of selected modules in the Navigation Subsystem are given limited autonomy, in order to eliminate potential bottleneck effects between the Pilot and the Naviga-

[1]The analogy confirms to the pilot concept introduced by A. Meystel in [11].

tion Subsystem. Furthermore, the Reactive Subsystem is invoked according to the strategy of the navigation context.

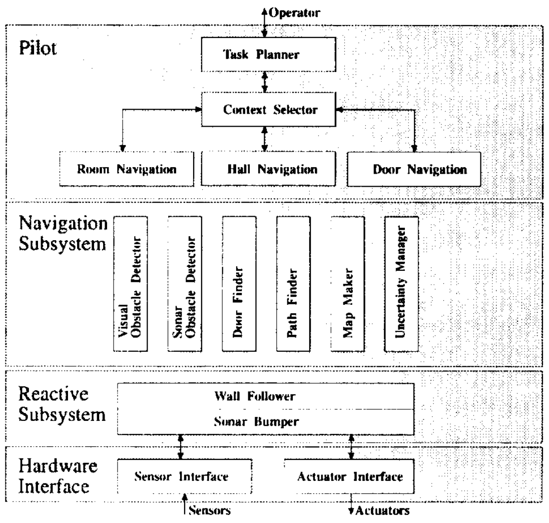

Figure 1: *The target architecture of the office delivery system.*

Room Navigation This context is selected by the Pilot for navigation in an office/laboratory room that may contain obstacles. The Pilot supplies the robot's initial position, with a bounded uncertainty and a goal position (in the same room) in the vicinity of the door to exit. The only a priori knowledge provided by the Pilot is the approximate position of the doors.

Given the start and goal points and a generated map, the Path Finder plans a collision-free path, which is updated incrementally upon detection of new obstacles. Obstacle (and free space) detection is based on ultrasonic sensors as well as stereo-vision, performed by the Sonar Obstacle Detector and the Visual Obstacle Detector, respectively. The coordinates of the detected obstacles are sent to the Map Maker, which builds and maintains a floor map. The Map Maker regularly transmits a copy of the updated map to the Path Finder, which consequently regenerates the path avoiding potential obstacles on the robot's path. The planned path is converted to a sequence of motion commands which are executed by modules at the hardware interface level. The executed motion commands are transferred to the Uncertainty Manager, which computes the positional uncertainty associated with the robot. The positional uncertainty is used by the Door Finder to determine the search area for finding doors.

During path execution the reactive subsystem ensures safe navigation using the Sonar Bumper, which acts as a safety agent with the objective to prevent the robot from colliding with static as well as moving obstacles. The Sonar Bumper reactively regulates the speed of the robot relative to the shortest distance to obstacles and eventually stops the robot if this distance becomes critically small.

Hall Navigation Hall navigation is similar to room navigation since a hall can principally be considered as a room, however, hallways/corridors have certain properties, which are exploited in order to achieve a more robust navigation. For this purpose a reactive wall following algorithm is employed, which ensures that the robot drives in the middle of the hallway and, at the same time, avoiding static obstacles.

Door Navigation For traversing the door the navigation system positions the door, approaches it and locates it again etc., until the robot traverses the door and enters a new room/hall. Positioning of the door is performed by a purposive visual module denoted the Door Finder.

3 System Modules

The system modules, are implemented in Vipwob (Vision Programmer's Workbench)[7]. Vipwob is a tool for distributed data processing across a computer network, which defines a high-level interface to networking. The modules communicate by *events* for low-level communication (e.g., control and synchronization), and *channels* for high band-width communication (e.g., for image transfer). A homogeneous module architecture contributes to the modularity of the system software. Furthermore, using events as the primary method of communication enables us to design a well-defined interface for each module. The logical architecture of the system modules is depicted in figure 2. The module has the following components:

Message handler: Handles incoming and outgoing traffic of events and thus defines the communication interface of the module. The message handler dispatches incoming events from the event queue and invokes the corresponding event-handler, which processes the event.

Services: The services correspond to event handlers and are invoked according to the incoming events.

User interface: The user interface is used for monitoring and interacting with the module.

Data interface: The data interface is used for high band-width communication.

Logically the events can be categorized in the following event types:

Simple events: consist of *signal*, *request* and *reply*. These events do not carry any explicit data, other than the semantics of the signal itself.

Data events: consist of *request* and *reply*. Data events contain data, and are used for information transfer from a source to a destination module.

The request and reply event types can be used to per-

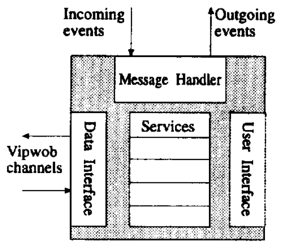

Figure 2: *Logical architecture of the system modules.*

form a hand-shake between the involved modules. Each of the modules included in the Navigation Subsystem, Reactive Subsystem, the Hardware Interface as well as the Pilot are implemented using this generic module architecture.

3.1 The Pilot

The Pilot is implemented as a rule-based production system in CLIPS[2] [5] embedded as a Vipwob module. The Pilot is principally divided into the following subsystems: *Task Planner*, *Context Selector*, *Navigation Modules* and the *Virtual Subsystem*. The Navigation Modules consist of *room*, *hall* and *door* navigation described in section 2.

The Task Planner, Context Selector and the Navigation Modules are implemented as CLIPS *modules*. It is used as an abstraction tool, as every construct defined by a module can not be 'seen' by other modules, unless explicitly specified. Each module has logically a working memory, and the contents of the working memory of a module is invisible to other modules. By partitioning the knowledge base into modules, a faster pattern-matching can be achieved, as only facts defined by the module are involved in the pattern-matching process.

The Virtual Subsystem of the Pilot consists of virtual modules, written in COOL (Clips Object-Oriented Language), which represent the real Vipwob modules in the Navigation and Reactive Subsystems. The virtual modules are implemented as classes and contain message-handlers corresponding to the services of each real Vipwob module. Furthermore, the classes contain fields to store data, which are used to store the internal state of the corresponding Vipwob module. When the Pilot requests some type of information, the particular object (i.e., instances of the classes at run time) can either forward the request to its corresponding Vipwob module, or it can return the information stored internally in the object, reducing the physical communication in certain cases. The Virtual Subsystem serves the purpose of hiding the communication specifics[3] to the system modules, so that the rule base of the Pilot is made more elegant.

Dividing the Pilot in the above subsystems, contributes to that the rule base is kept in a modular manner. Augmenting the rule base can be done relatively easy as the Pilot is modular and knowledge is embedded in rules.

3.2 Path Finder

The path planning method used in AMOR is based on a potential-field [6, 10] method, for safe path generation. One of the major disadvantages of the potential field methods, however, is that the usual formulations of potential-field path planning do not preclude the occurrence of minima other than the goal. In the steepest-decent search the robot can 'fall' into these minima and achieve a stable configuration short of the goal state. In [2] Connolly et al. describe the application of *harmonic functions* to the potential-field path-planning problem. Harmonic functions are shown to have several useful properties, one of which is that creation of local minima within the solution region is impossible. Furthermore harmonic functions allow incremental update of the path, resulting in reactive planning[4], which makes them suited for adaptive regeneration of paths in unknown configurations.

3.3 Map Maker

The Map Maker is based on a slightly modified version of *histogram grid* methods [1, 13]. The modification consist of assigning a size/ uncertainty to obstacles and thus updating a set of cells in the grid, instead of only updating one cell. The obstacle detectors have a constant sampling frequency, and therefore the region covered by the sensors (or the spatial resolution) will depend upon the travelling speed of the robot. To obtain a map without holes, it is required to scale the size of detections according to the travelling speed and sampling frequency of the obstacle detectors[5].

The purpose of the Map Maker is to build a consistent map of the environment by filtering out spurious detections and keeping correct detections in the map. This is obtained by setting a threshold on the number of detections required before an obstacle appears on the map. Currently, the threshold is set to 3 detections at the same location, which gives a consistent map. However, it appears that this threshold might result in reduced reactivity to new obstacles. Higher reactivity is obtained by including obstacles in the map, the first time they are detected, but reoccuring detections are required to include the obstacle for permanent inclusion in the map. This scheme results in a map with few mis-detections and high reactivity.

3.4 Uncertainty Manager

The Uncertainty Manager computes the positional uncertainty associated with the robot, specified by a mean

[2](C Language Integrated Production System.)

[3]CLIPS has in this project been augmented with user-defined functions, which handle communication to the real Vipwob modules.

[4]In our implementation path replanning consumes a time of approximately 4 seconds on a MIPS 3000 CPU.

[5]In the case of $v = 0.1m/s$ and $freq = 1Hz$, the size will be $0.1m$.

value and a covariance matrix. Based on this positional uncertainty the Door Finder determines where in the image to search for the door. The method used for computation of the positional uncertainty is based on a method developed by Kosaka and Kak described in [8].

A plant (vehicle) model is developed for the mobile platform separated in a model for pure translation and pure rotation, respectively. The motion of the robot is defined with a number of random variables (with assumed Gaussian distribution), which characterize the actual motion the robot undergoes as a function of a given motion command (i.e., different distances for pure translation and different rotations for pure rotation). The vehicle model is then used to update the positional uncertainty of the robot as new motion commands are executed.

3.5 Door Finder

The Door Finder is a purposive visual module which uses three features for finding the door. The door finder assumes that the door posts are ***two straight lines***, which are detected with a robust edge detection method optimized for detecting straight lines. The approximate ***position*** of the door is used to determine the image processing area by transforming the world coordinates of the door, with a given uncertainty[6], to the image plane. The *width of the doorway*, i.e., the distance between the door posts is known with a small constant uncertainty, and is therefore used to filter out erroneous responses from the two previous steps.

The edge detector operates in five steps: the image is convoluted with an simple 1 × 3 ([-1,0,1]) differentiating kernel, combined with a Hough transform which is restricted to almost vertical lines. The thresholded gradient images is subsequently AND'ed with the longest lines from the Hough transform. The resulting lines are linked, and the image coordinates of the lower end-points of the lines determine where the door post intersect the ground plane. Subsequently the world coordinates of the intersections are computed using inverse perspective transformation from the image plane to the ground plane.

Two sample images produced by the Door Finder are shown in figure 3. The white areas in the images correspond to the thresholded response from the gradient kernel. Several (almost) vertical lines (black lines) are found in both images, but the correct door coordinates are selected by criteria on the estimated position and door width.

The largest deviation allowed, when finding the door, for successful door traversing is approximately $150mm$[7]. The accuracy of the Door Finder with a detection distance of $2.5m$ is maximum $45mm$ and the mean error is $37mm$, which is well inside the required range. The test results are shown in figure 4.

[6]Provided by the Uncertainty Manager.

[7]The width of the robot is about $700mm$ and the door opening is about $1000mm$ wide.

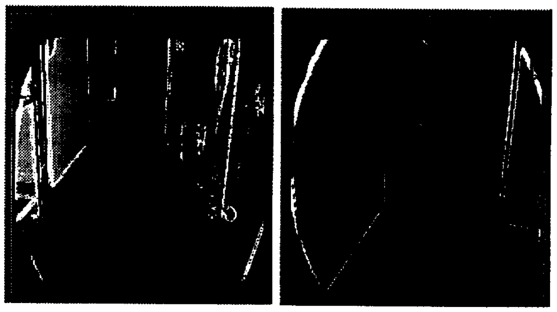

Figure 3: *Sample images produced by the Door Finder. The left image is from a distance of approximately* $4.5m$, *and the right image is at a distance of approximately* $2.5m$.

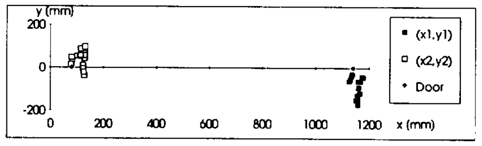

Figure 4: *The Door Finder results from the distance* $2.5m$ *to* $4.7m$. *The resolution of the axes is equal to give a geometrical impression of the accuracy of the detection.*

3.6 Obstacle Detector

The visual obstacle detects obstacles by inverse perspective transformation from the image plane to the ground plane of a pair of stereo images, followed by an absolute subtraction of the two transformed images [9, 3]. The obstacle detection is obtained by thresholding the difference image, and subsequently filtering it with a morphological erosion. The remaining pixels in the image are labeled as obstacles, and the pixels filtered out, are labeled as free-space.

The advantage of this method is that the coordinates of an obstacle are given directly from the (x, y) coordinates in the ground plane image (as a result of the transformation to the ground plane), and that the method classifies objects deviating from the ground plane. No object recognition or high level segmentation is required for this method to work.

With an image resolution of 256 × 256 and the used camera position, the resolution of the ground plane image is better than $40mm$. The Obstacle Detector showed an error which was linearly dependent on the distance. The source of the linear error is that obstacles are detected at a given height[8], which in our case was about $200mm$. The linear error is found by fitting a line to the test results by linear regression, giving the slope of the error. The slope is then used to compensate for the error resulting in a standard error of $80mm$ in distance to the detected obstacles. The field of view of the vi-

[8]This depends upon the camera set-up and the image resolution [14].

sual Obstacle Detector extends from 1.5m to 4.3m with a width of approximately 2.8m. The sampling frequency of the Obstacle Detector is approximately 1Hz and this is comparable to the sampling frequency of the sonars.

3.7 Sonar Bumper

The Sonar Bumper ensures safe collision-free navigation by reactively regulating the speed of the robot based on the minimum distance to obstacles. It is based on 24 ultrasonic sensors, which directly provide range information, thus the processing requirements of the Sonar Bumper are low, resulting in a tight coupling between sensing and action.

The Sonar Bumper is capable of stopping the vehicle with an accuracy of about $\pm 50mm$ from a specified minimum distance from obstacles, i.e., for minimum distance set to $> 100mm$, the Sonar Bumper guarantees collision free navigation.

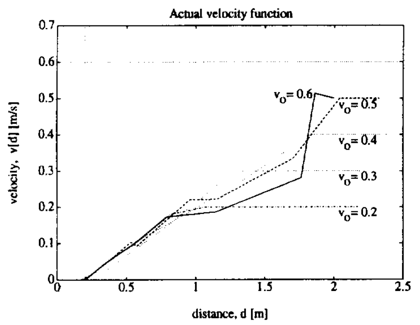

Figure 5: *The actual speed of the robot, $v[d]$, (at discrete time instances) as a function of the robots distance to some obstacle, d. The vertical line, $d = 0.2$, is the safety distance where the vehicle should be at total rest.*

Figure 5 depicts the way the Sonar Bumper affects the robot's speed with decreasing distances to obstacles. The plots in the figure are for different initial velocities (for $v_0 = 0.2, 0.3, 0.4, 0.5$ and $0.6m/s$). From the plot for $v_0 = 0.6m/s$, it can be seen that the first time a sample of the sonars is retrieved the speed is reduced drastically (before the robot has accelerated to the initial speed), which indicates that an initial speed of $0.6m/s$ is not sensible for the given system.

3.8 Wall Follower

In the hall navigation context, the Wall Follower ensures collision free navigation by guiding the robot to the center of the hallway and at the same time avoiding collisions. The Wall Follower reacts to the sonar range readings provided by the sonars mounted on the left and right sides of the robot. The sonar data are transformed to a robot-centered coordinate system and subsequently two straight lines, l_l and l_r, are fitted to the range data retrieved from the sonars on respectively the left and right sides of the robot[9]. The lines represent repulsive forces with magnitude as the reciprocal of the robot's distance to the fitted lines, with direction given by the normal of the lines. Thus the repulsive force increases as the robot approaches a wall.

[9]There are 9 sonars on each side of the robot.

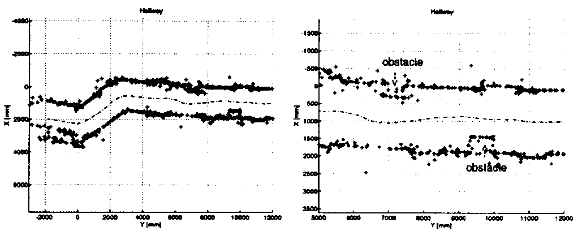

Figure 6: *The walls, as seen by the robot (dots), are not straight, rather the plot depicts how the robot perceives the hallway. (left) Note how the robot follows the walls at the same time avoiding obstacles. (right) Zoom in on the area where the robot avoids some obstacles.*

The sum of these forces, and the direction of the robot, gives the direction followed by the robot. Since the lines are fitted to local data, the robot is capable of avoiding potential obstacles planted in the hallway. Figure 6 depicts a plot of the robot's path in the hallway, where the robot is guided by the Wall Follower. It can be seen that the robot moves in the middle of the hallway and that the path follows the bends of the hallway. Additionally, it can be seen (right plot) that the Wall Follower enables the robot to avoid obstacles, without explicit obstacle detection.

4 Experimental Results

In the following, results from test runs from a navigation session from one room to another, will be presented. The room navigation tests were performed at the Laboratory of Image Analysis (LIA) and the robot was given the task to autonomously navigate from a known start position to a specified goal position. The only a priori knowledge of the robot is the size of the room, and the approximate position of the doors. The robot speed for the test run is $\leq 0.1m/s$ and the path is registered by sampling the odometry with a frequency of $1Hz$. The obstacles are mostly static.

Figure 7 (left) depicts one of the tests, where the robot navigates continuously[10] from start to goal. The chequered patterns are plots of the obstacles detected by the robot and are roughly consistent with the actual objects scattered in the room, as the occupied places indicated in figure 7 usually contain desks, cabinets etc. The generated path is collision-free and the average time for completing the navigation task is about 3 minutes for each room.

Results of Room to Room Navigation The room to room navigation test consisted of positioning the robot

[10]The dashed curve in the figure.

in room D1-105 and command it to go to another room (D1-101).

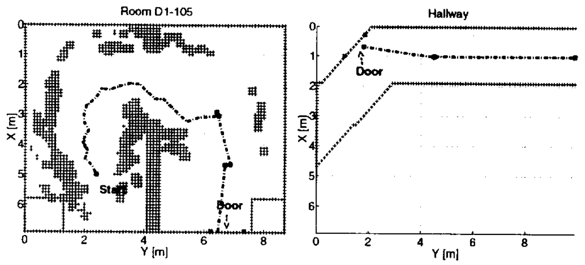

Figure 7: *(left) Going from D1-105 to the hall. The robot performs three sub-paths, going to vicinity of the door, finding the door, approaching it and finally going through it. (right) After entering the hall, from room D1-105, the robot goes through the hall (no obstacles were placed in the hall) and enters the next room. The average time for completing the task was about 5 min.*

Figure 7 depicts the path of the robot including when navigating in the initial room. The path consists of three segments. The first path brings the robot to the vicinity of the door (to exit), then the door is found and the robot approaches the door, while aligning itself for safe door traversing. Subsequently, the door is found again with higher precision, the door is repositioned and then the robot goes through the door. In the figure it seems like the robot does not exit the door optimally, because the path goes to the right. This is, however, not the fact. What happens here is that, the robot has an accumulated position error due to wheel slippage. When it finds the door, it will not be in the expected world coordinates due to this position error. AMOR will thus reposition the door relative to its own position to enable safe door traversing. In figure 7 (right) the robot has entered the hall (at the top-right) and heads for room D1-101. The robot uses the same strategy for going through the door and finally it enters the room (not depicted here) and accomplishes the navigation task successfully.

5 Summary

In this paper an autonomous mobile robot navigation system, capable of performing automated office delivery tasks, is described. The implemented prototype system is capable of autonomously navigating from one room to another designated by the operator. This involves: navigation in a room, where the robot dynamically detects and avoid obstacles; hall navigation, where it follows the walls by a reactive scheme and finally door traversing, where the robot autonomously positions the door and passes safely through it. The only a priori knowledge used by the system consists of a graph representation of the navigation environment, containing door positions. A priori, no knowledge is provided about the position, physical or geometrical properties of other objects in the environment.

The results indicate that using visual capabilities enables robust goal directed mobile robot navigation systems guided by a set of carefully selected visual modules. Additionally reactive systems based on simple sensing modalities enable safe navigation and handling of run-time contingencies.

References

[1] Johann Borenstein and Yoram Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):S278-288, 1991.

[2] Christopher I. Connolly and Roderic A. Grupen. On the application of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):pp 931-946, October 1993.

[3] Henrik I. Christensen Niels O. S. Kirkeby et al. Active vision for mobile robot navigation. *Proceedings Intelligent Robotic Systems '93 Zakopane*, pages 44-56, July 1993.

[4] I. Nourbacksh; S. Morse et al. The winning robots from the 1993 robot competition. *AI Magazine*, 14(4):51-66, Winter 1993.

[5] Joseph C. Giarratano. *Expert Systems, - Principles and Programming.* PWS-Kent Publishing Company Boston, 1 edition, 1989.

[6] O. Khatib. Commande Dynamique dans l'Espace Operationnel des Robots Manipulateurs en Presence d'Obstacles. Docteur Ingenieur Thesis, L'Ecole Nationale Superieure de l'Aeronautique et de l'Espace, Toulouse, France, 1980.

[7] Niels O. S. Kirkeby and Henrik I. Christensen. Vipwob 4.0 reference manual. *Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University*, February 1993.

[8] Akio Kosaka and Avinash C. Kak. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *CVGIP: Image Understanding*, 56(3):271-329, November 1992.

[9] Jana Kosecka and Ruzen Bajcsy. Discrete event systems for autonomous mobile agents. *Proceedings Intelligent Robotic Systems '93 Zakopane*, pages 21-31, July 1993.

[10] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[11] A. Meystel. *Autonomous Mobile Robots*, volume 1. World Scientific Series in Automation, 1991.

[12] Paolo Pirjanian and Hans Blaasvær. AMOR: An Autonomous Mobile Robot Navigation System. Master's Thesis, Laboratory of Image Analysis, Institute of Electronic Systems, Aalborg University, Denmark, 6 1994.

[13] Steven Ratering and Maria Gini. Robot navigation in a known environment with unknown moving obstacles. *1993 IEEE International Conference on robotics and Automation*, 3:25-30, 1993.

[14] Cordelia Schmid and Philippe Bobet. Obstacle detection analysis. *SMART Workshop, ISPRA Italy*, April 1994.