ViRbot: A System for the Operation of Mobile Robots

Conference Paper · May 2007

DOI: 10.1007/978-3-540-68847-1_55 · Source: DBLP

7 authors, including:

Jesus Savage
Universidad Nacional Autónoma de México
67 PUBLICATIONS **343** CITATIONS
SEE PROFILE

Adalberto Llarena
Universidad La Salle México
17 PUBLICATIONS **100** CITATIONS
SEE PROFILE

Sergio Cuellar
Pontificia Universidad Javeriana
2 PUBLICATIONS **20** CITATIONS
SEE PROFILE

Ulises Penuelas
Universidad Nacional Autónoma de México
3 PUBLICATIONS **17** CITATIONS
SEE PROFILE

ViRbot: A SYSTEM FOR THE OPERATION OF MOBILE ROBOTS

Jesus Savage, Adalberto LLarena, Gerardo Carrera, Sergio Cuellar, David Esparza, Yukihiro Minami, Ulises Peñuelas
Bio-Robotics Laboratory, Department of Electrical Engineering
Universidad Nacional Autónoma de México, UNAM
Email: savage@servidor.unam.mx

***Abstract*— This paper describes a robotics architecture, the ViRbot, used to control the operation of service mobile robots. It accomplish the required commands using AI actions planning and reactive behaviors with a description of the working environment. In the ViRbot architecture the actions planner module uses Conceptual Dependency (CD) primitives as the base for representing the problem domain. After a command is spoken to the mobile robot a CD representation of it is generated, a rule based system takes this CD representation, and using the state of the environment generates other subtasks represented by CDs to accomplish the command. By using a good representation of the problem domain through CDs and a rule based system as an inference engine, the operation of the robot becomes a more tractable problem and easier to implement.**

I. INTRODUCTION

There is a need for reliable Human-Robot interaction systems as experienced by the proliferation of new humanoid robots in particular in Japan with advanced interaction capabilities including spoken language. This can also be appreciated by new competitions to exalt the robots social interaction with humans such as the new league in the robots' competition Robocup: RoboCup@Home. The goal of this league is to promote the development of real-world applications and human-machine interaction with autonomous robots, or as the competition organizers put it: "The aim is to foster the development of useful robotic applications that can assist humans in everyday life" [1].

In this paper is presented a mobile robot architecture, the ViRbot system, whose goal is to operate autonomous robots that can carry out daily service jobs in houses, offices and factories.

The ViRbot system was tested in the Robocup@Home category in the Robocup competition[1] at Bremen, Germany in 2006; it will be used again in the same competition in Atlanta, 2007, with the robots TX8 and TPR8, see figure 1.

The ViRbot system [2] divides the operation of a mobile robot in several subsystems, see figure 2. Each subsystem has a specific function that contributes to the final operation of the robot.

Each of the layers of figure 2 will be described in the following sections.

II. VIRTUAL ENVIRONMENT

The virtual environment is visualized by a 3D system called ROC2. The simulation environment can be changed easily; it has multiple view ports of the simulation; it has local or remote interaction (Internet); it can execute an user's subroutines written in C/C++; simulation of the robot's movements and sensor's readings can be provided also by the user using C/C++, see figure 3.

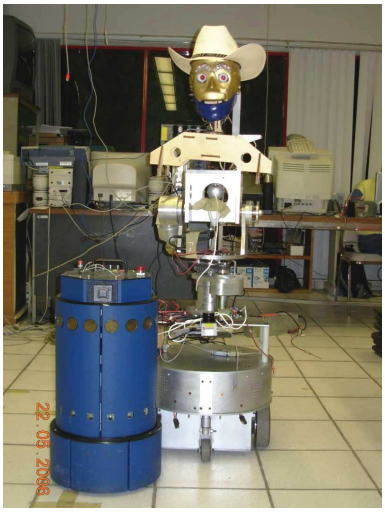

Fig. 1. Testing Robots TX8 and TPR8, these robots have, for sensing the environment, a ring of sonars, a laser measurement system, a microphone and a vision system.

III. INTERNAL SENSORS

The robots used in this research have the following internal sensors that reflect its internal state: wheel encoders and battery level sensor. Each of the sensors' values can be read any time during the operation of the robot.

IV. EXTERNAL SENSORS

The robots have the following external sensors that sense the surrounding environment: contact, reflective, infrared, video-cameras, microphones. Digital signal processing techniques are applied to the signals obtained from the video-cameras and microphones to be used in pattern recognition algorithms.

V. SIMULATOR

In the development of a complex service robot system it is necessary to test different algorithms and procedures, to do this

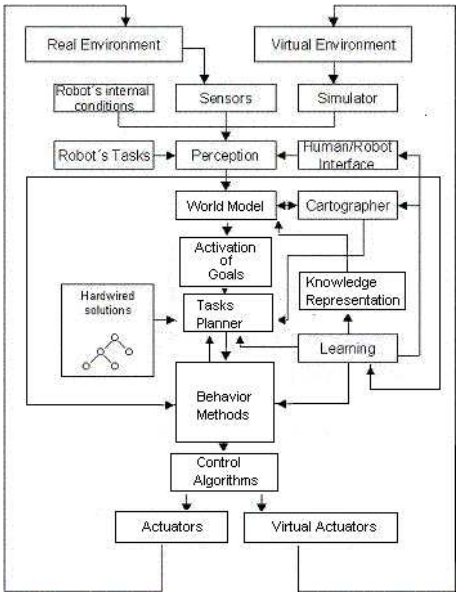

Fig. 2. The ViRbot System consists of several subsystems that control the operation of a mobile robot.

in a real robot it could be very expensive and time consuming, and in some cases it could put the robots in danger and also the people that surround them. Thus it is necessary to have a simulator system where the algorithms can be tested first in the virtual robot and then in the real one.

The ViRbot system contains a simulator that provides the values of the internal and external sensors that the robot has, each simulated sensor has a mathematical model. Also the simulation of new sensors can be incorporated easily.

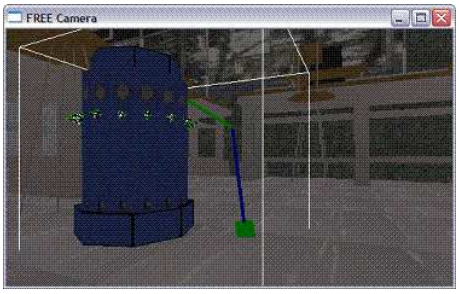

Fig. 3. In this figure it is shown the simulation of the robot TX8 using the 3D system ROC2

VI. Robot's Tasks

Set of tasks that the robot needs to accomplish according to the time when it was programmed.

Example:

Deliver item X to place I at time Tm
.
.
.
Deliver item Y to place J at time Tn

VII. Human/Robot Interface

The Human/Robot Interface subsystem in the ViRbot architecture has tree modules: Natural Language Understanding, Speech Generation and Robot's Facial Expressions.

A. Natural Language Understanding

The natural language understanding module finds a symbolic representation of spoken commands given to a robot.

It consists of a speech recognition system coupled with Conceptual Dependency techniques [3].

1) Speech Recognition: For the speech recognition system it was used the Microsoft Speech SDK engine. One of the advantage of this speech recognition system is that it accepts continuous speech without training, also it is freely available and with C++ examples that can be customized as it is required. It allows the use of grammars, that are specified using XML notation, which constrains the sentences that can be uttered and with that feature the number of recognition errors is reduced. Using a grammar, the transitions from one word to another is restricted, reducing the perplexity considerable. Almost every existing speech recognition system has a major problem, the insertion words, words incorrectly added by the speech recognition system. These words may cause a robot to fail to perform the asked command, then it is necessary to find a mechanism that, even if these errors exists, the robot should be able to perform the required commands.

2) Conceptual Dependency: One way to represent a spoken command is by describing the relationships of objects mentioned in the input sentence. During this process, the main event and the participants, that are described in the sentence are found. In this work the participants are any actors and recipients of the actions. The roles the participants play in the event are determined, as the conditions under which the event took place. The key verb in the sentence can be used to associate the structure to be filled by the event participants, objects, actions, and the relationship between them.

Conceptual Dependency is a theory developed by Schank for representing meaning. This technique finds the structure and meaning of a sentence in a single step. CDs are especially useful when there is not a strict sentence grammar.

One of the main advantages of CDs is that they allow rule based systems to make inferences from a natural language system in the same way humans beings do. CDs facilitate the use of inference rules because many inferences are already contained in the representation itself. The CD representation uses conceptual primitives and not the actual words contained in the sentence. These primitives represent thoughts, actions, and the relationships between them.

Some of the more commonly used CD primitives are, as defined by Schank:

ATRANS: Transfer of ownership, possession, or control of an object (e.g. give.)

PTRANS: Transfer of the physical location of an object (e.g. go.)

ATTEND: Focus a sense organ (e.g. point.)

MOVE: Movement of a body part by its owner (e.g. kick.)
GRASP: Grasping of an object by an actor (e.g. take.)
PROPEL: The application of a physical force to an object (e.g. push.)
SPEAK: Production of sounds (e.g. say.)

Each action primitive represents several verbs which have similar meaning. For instance give, buy, and take have the same representation, i.e., the transference of an object from one entity to another.

Each primitive is represented by a set of rules and a data structure containing the following categories, in which the sentence components are classified:

An Actor: The entity that performs the ACT.
An ACT: Performed by the actor, done to an object.
An Object: The entity the action is performed on.
A Direction: The location that an ACT is directed towards.

The user's spoken input is converted into a CD representation using a two step process. The CDs are formed first by finding the main verb in the spoken sentence and choosing the CD primitive associated with that verb. Once the CD primitive has been chosen the other components of the sentence are used to fill the CD structure.

For example, in the sentence "Robot, go to the kitchen", when the verb "go" is found, a PTRANS structure is issued. PTRANS encodes the transfer of the physical location of an object, and it has the following representation:

(PTRANS (ACTOR NIL) (OBJECT NIL) (FROM NIL) (TO NIL))

The empty (NIL) slots are filled by finding relevant elements in the sentence. So the actor is the robot, the object is the robot (meaning that the robot is moving itself), and the robot will go from the living room to the kitchen (assuming the robot was initially in the living room). The final PTRANS representation is:

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM living-room) (TO kitchen))

Another example, the phrase: John gave the book to Mary, can be represented by the following CD:

(ATRANS (ACTOR John) (OBJECT book) (FROM John) (TO Mary))

There are also primitives that represent the state in which an object is, for the previous example the book now belongs to Mary is represented by:

(POSSESS (OBJECT Mary) VALUE Book))

CD structures facilitate the inference process, by reducing the large number of possible inputs into a small number of actions. The final CDs encode the users commands to the robot. To carry out these commands an actions planning module must be used as described in the following section. CDs are suitable for the representation of commands and for asking simple questions to a robot, but they are not suitable for the representation of complex sentences.

B. Speech Generation and Robot's Facial Expressions

For a truly Human-Robot interaction a robot needs to answer questions or establish a dialog also through the generation of synthetic speech. There are several text to speech systems available, in order to use these systems in an appropriated way, the text that is spoken needs to be generated according to the context surrounding a situation. The rule based system generates the text to be spoken through the CD primitive SPEAK.

This CD representation for generating speech has the following components:

(SPEAK (ACTOR Robot) (OBJECT "Sentence to be spoken") (TO User))

Using this terminology the Robot is the one that speaks the required sentence to the user.

The text to speech generation system used is called Festival [4] that is freely available.

In human to human communication, facial expression plays an important role, so we consider that the same thing applies to human to robot communication. Thus, one of our robots contains a mechatronic head that shows simple expressions through movements of it, the opening of its mouth, the movement of its eyes and by modifying its eyebrows. The eyebrows are created using an array of LEDs that are turned on and off that creates different face expressions, see figure 4.

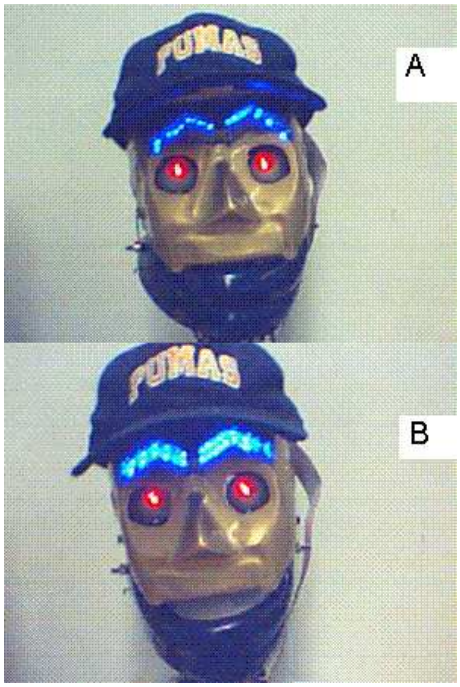

Fig. 4. In A, the robot's face shows that the robot did not understand the given command; in B, the robot's face shows that it understood the spoken command and that it is ready to execute it.

When the expression of the face changes the robot is ready for accepting commands, when it understood them and also when is executing them, etc.

VIII. Perception

The perception module obtains a symbolic representation of the data coming from the users and the robots sensors. The symbolic representation is generated by applying digital processing algorithms on the data coming from the sensors. With this symbolic representation a belief is generated.

A. Vision Subsystem

The vision subsystem consists of a robust implementation of an object tracker using a vision system that takes in consideration partial occlusions, rotation and scale for a variety of different objects. The objects are represented by feature points which are described in a multi-resolution framework, that gives a representation of the points in different scales. The interest points are detected using the Harris detector [5], and the description was based in an approximation coined SURF (Speeded-Up Robust Features)[6].

A stack of descriptors is built only the first time that the interest points are detected and extracted from the object previously selected. This action adds a faster computation due to the fact that it can be done off-line. The object is tracked by an Unscented Kalman Filter (UKF) [7] using a constant velocity model that estimates the position and the scale of the object, with the uncertainty in the position and the scale obtained by the UKF, the search of the object can be constrained only in an specific region from the whole image. Thus, the match of the points is performed only in a small part of the stack of descriptors. The use of this approach showed an improvement to real-time motion tracking and to recover from full occlusions. The approach does not assume the form of the object and the results shown that it can track successfully and efficiently identified objects, see figure 5.

Fig. 5. The vision subsystem tracks and identifies objects, taking in consideration partial occlusions of the objects as well as their rotations and changes in scale.

IX. Cartographer

This module has different types of maps for the representation of the environment:

Raw maps are obtained by detecting the position of the obstacles using the robot's sonar and laser sensors .

Symbolic maps contain each of the known obstacles defined as polygons, that consists of a clockwise ordered list of its vertexes. Forbidden areas are areas which are not allowed for the robot to enter, they are built by growing the polygons that represent the objects by a distance greater than the radius of the robot, to consider it as a point and not as a dimensioned object. It is possible to create the configuration space in this way because our robot has cylindrical shape.

The Cartographer subsystem contains also topological and probabilistic (Hidden Markov Model) maps of the environment [8], see figure 6.

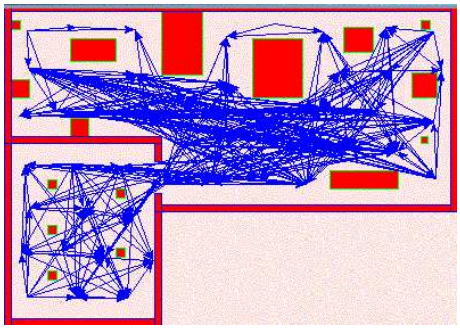

Fig. 6. The Cartographer subsystem has a topological map of the environment in the free space; the rectangles in the figure represent the known fixed obstacles in the environment.

X. Knowledge Representation

A rule based system is used to represent the robot's knowledge, that is represented by rules, each one contains the encoded knowledge of an expert. In the ViRbot the rule based system CLIPS is used, an expert system shell, freely available and developed by NASA [9].

XI. World Model and Activations of Goals

The belief generated by the perception module is validated by the cartographer and the knowledge representation modules, thus a situation recognition is created. Given a situation recognition, a set of goals are activated in order to solve it.

XII. Hardwired Solutions

Set of hardwired procedures that solve, partially, specific problems. Procedures for movement, transference of objects, pick up, etc.

XIII. Task Planner

After receiving the CD representation from the Human/Robot interface the Perception subsystem perceives a new situation that needs to be validated by the World Model subsystem. The World Model validates the situation by the information provided by the Cartographer and the Knowledge Representation subsystem. The Planner subsystem takes as an input the output of the World Model subsystem and tries to take care of the situation presented. Planning is defined as the

process of finding a procedure or guide for accomplishing an objective or task. In the ViRbot planner subsystem there are two planning layers, the upper is the actions planner, based on a rule based system, and the lower layer the movements planner, based on the Dijkstra algorithm.

A. Actions Planner

The Robot is able to perform operations like grasping an object, moving itself from on place to another, etc. Then the objective of action planning is to find a sequence of physical operations to achieve the desired goal. These operations can be represented by a state-space graph. Thus, action planning requires searching in a state-space of configurations to find a set of the operations that will solve a specific problem. Internally, the actions Planner is built using CLIPS, that has an inference engine and this uses forward state-space search, that finds a sequence of steps that leads to a solution given a particular problem. Actions planning works well when there is a detailed representation of the problem domain. In the ViRbot architecture the actions planning module uses conceptual dependency as the base for representing the problem domain. After a command is spoken, a CD representation of it is generated. The rule based system takes the CD representation, and using the state of the environment it will generate other subtasks represented by CDs and micro-instructions to accomplish the command. The micro-instructions are primitive operations acting directly on the environment, such as operations for moving objects.

For example when the user says **"Robot, go to the kitchen"**, the following CD is generated:

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM Robot's-place) (TO Kitchen))

It is important to notice that the user could say more words in the sentence, like **"Please Robot, go to the kitchen now, as fast as you can"** and the CD representation would be the same. That is, there is a transformation of several possible sentences to a one representation that is more suitable to be used by an actions planner. All the information required for the actions planner to perform its operation is contained in the CD.

B. Movements Planner

If the command asks the robot to go from one room to another the movements planner finds the best sequence of movements between rooms until it reaches the final destination. Inside of each room the movements planner finds also the best movement path considering the known obstacles, that represent some of the objects in the room. Thus, the movements planner uses this information to find the best path avoiding the obstacles that interfere with the goal. The best path is found among several paths, according to some optimization criteria and using the Dijkstra algorithm. For the previous example where the robot was asked to go to the kitchen, the movements planner just needs to find the best global path between the Robot's place and the Kitchen, thus the action planner issues the following command to the movements planner:

(MOVEMENTS-PLANNER get-best-global-path Robot's-place to Kitchen)

And the answer of the movements planner is the following:

(best-global-path $Robot's - place\ place_1\ place_2 ... place_n\ Kitchen$)

Now a new set of PTRANS are generated asking the robot to move to each of the places issued by the planner:

(PTRANS (ACTOR Robot) (OBJECT Robot) (FROM $place_i$) (TO $place_j$))

For each of these PTRANS it is asked to the movements planner to find the best local path from $place_i$ to $place_j$:

(MOVEMENTS-PLANNER get-best-local-path $place_i$ to $place_j$)

And the answer of the movement planner is the following:

(best-local-path $node_1\ node_2 ... node_m$)

in which each $node_i$ is part of the topological map whose coordinates are x_i and y_i that the robot needs to reach, see figure 7.

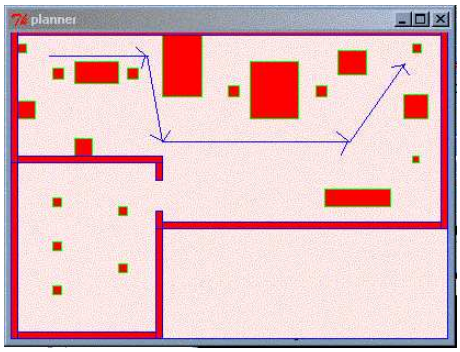

Fig. 7. The movements planner uses a topological map of the room, which takes into account the objects in each of the rooms, and it finds a local path with the Dijkstra algorithm.

XIV. Behavior Methods

After the movements planner finds the nodes where the robot needs to go, the Behavior subsystem tries to reach each of them, if it finds unexpected obstacles during this process it avoids them. The Behavior subsystem consists of behaviors based on potential fields methods and state machines, all these controls the final movement of the robot [10].

A. Potential Fields

Under this idea, the robot is considered as a particle that it is under the influence of an artificial potential field $\vec{U}$ whose local variations reflects the free space structure and it depends on the obstacles and the goal point that the robot needs to reach [11]. The potential field function is defined as the sum of an attraction field that push the robot to the goal and a repulsive field that take it away from the obstacles. The movement

planning is done by iterations, in which and artificial force is induced by

$$\vec{F}(q) = -\vec{\nabla}\vec{U}(q) \quad (1)$$

that forces the robot to move to the direction that the potential field decrees, where $\vec{\nabla}$ is the gradient in q and $q = (x, y)$ represents the coordinates of the robot position. The potential field is generated by adding the attraction field $\vec{U}_{atr}$ and the repulsive filed $\vec{U}_{rep}$

$$\vec{U}(q) = \vec{U}_{atr}(q) + \vec{U}_{rep}(q)$$

thus

$$\vec{F}(q) = \vec{F}_{atr}(q) + \vec{F}_{rep}(q)$$

where

$$\vec{F}_{atr}(q) = -\vec{\nabla}\vec{U}_{atr}(q)$$
$$\vec{F}_{rep}(q) = -\vec{\nabla}\vec{U}_{rep}(q)$$

Defining the unitarian vector pointing to the gradient direction

$$\vec{f}(q) = \frac{\vec{F}(q)}{\|\vec{F}(q)\|}$$

in this way the movement in discrete times is defined by

$$q_{i+1} = q_i + \delta_i \vec{f}(q) \quad (2)$$

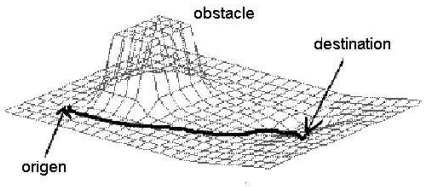

Fig. 8. Path found using potential field

XV. Control Algorithms and Real and Virtual Actuators

Control algorithms, like PID, are used to control the operation of the virtual and real motors. The virtual or the real robot receives the commands and executes them by interacting with the virtual or real environment and with the user.

XVI. Learning

The following learning algorithms are used for our robots:

a) *Movements of the robots.*- An Artificial Neural Feed Forward Network (ANFFN) was used to teach the robots to follow walls, by putting sonar readings in the inputs of the ANFFN and its outputs control the movement of the robot.

b) *Map building.*- Fuzzy Clustering Networks are used to locate and represent obstacles in the environment.

c) *Self Localization.*- Hidden Markov Models using Vector Quantization & Self Associative Neural Networks are used to find an estimation of the robots' position and orientation.

The robot are capable of generating training sets in real-time and creating new ANFFN. As long as they move in the environment, all the sensor readings are being stored for further analysis. The architecture of ViRbot is intended to facilitate the automatic learning of new tasks instead of spending large amounts of time with tedious programming hours.

XVII. Conclusions and Discussion

The ViRbot system was tested on how well the robot performed orders of the type: "Robot, go with the Father", "Robot, go to the kitchen", "Robot, give the newspaper to the Mother", "Robot, where is the newspaper?", etc. The environment represented a house, set in our laboratory, based on the layout proposed by the Robocup@Home competition[1], see fig 7. Conceptual Dependency representation helped to increase the speech recognition rate and also helped the actions planner to perform the desired user's command. The use of Conceptual Dependency to represent spoken command given to a robot helped to the actions planner in the ViRbot system to find a set of steps to accomplish the required commands. Conceptual Dependency is useful to represent basic commands and simple dialog with a robot. Some videos showing the operation of the Virbot system can be seen in *htpp://biorobotics.fi-p.unam.mx.*

References

[1] Robocup@Home, http://www.ai.rug.nl/robocupathome/, 2006.

[2] J. Savage, M. Billinhurst, A. Holden. The ViRbot: a virtual reality robot driven with multimodal commands. *Expert Systems with Applications 15, 413-419, Pergamon Press, 1998.*

[3] Roger C. Schank. Conceptual Information Processing. *North-Holland Publishing Company, 1975.*

[4] http://www.cstr.ed.ac.uk/projects/festival/

[5] Harris C. J. and Stephens M., A combined corner and edge detector. *pp 147-171, 1988. Proc. 4th Alvey Vision Conferences.*

[6] Bay H, Tuytelaars T, and Van Gool L., SURF: speed up robust deatures. *May 2006. Proceedings of the ninth European Confernce on Computer Vision.*

[7] Julier S.J and Uhlmann J.K. Uncented filtering and nonlinear estimation. *Volume 93, pp 401-422. Proceedings of the IEEE, 2004.*

[8] Thrun, Sebastian, and Bucken, A. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. *In Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Pres, 1996.*

[9] CLIPS Reference Manual Version 6.0. Technical Report Number JSC-25012. *Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX, 1994.*

[10] Muller J.P. and M.Pischeli, and M. Thiel. Modeling reactive behavior in vertically layered agent architectures *Intelligent Agents - Theories, Architectures, and Languages, volume 890 of Lecture Notes in AI. Springer, January 1995.*

[11] J.-C. Latombe, Robot Motion Planning. *Massachussets, USA: Kluwer Academic Publishers, 1991.*