In Proc. of the 12th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2003, St. Louis, USA, May 25-28, 2003, pp. 266-271

# Autonomous Navigation in a Known Dynamic Environment

László Kiss[1,2], Annamária R. Várkonyi-Kóczy[1,2], Péter Baranyi[2,3]

[1]Department of Measurement and Information Systems

[2]Integrated Intelligent Systems Japanese-Hungarian Laboratory

[3]Department of Telecommunications and Telematics

Budapest University of Technology and Economics
Magyar tudósok körútja 2., H-1117 Budapest, Hungary
{kissl,koczy}@mit.bme.hu, baranyi@alpha.ttt.bme.hu

*Abstract*—**This paper proposes a method for the navigation of an autonomous robot used inside a building. The main problem to be solved in this issue is the need to use both a priori information on the environment and momentary sensor data in a sensible way, and thus make the robot capable of finding its way around while also avoiding the obstacles, both static and dynamic. In the paper, a hybrid navigation method is proposed, using two techniques that deal with a priori information and sensory data separately. An algorithm for finding the optimal route to the goal using a priori information is suggested. The properties of the complete system are verified by computer simulation.**

**Keywords: mobile robots, local navigation, global navigation, generalized fuzzy-neural network, A* algorithm, potential field based guiding**

## I. Introduction

Autonomous navigation is nowadays a popular field of research all over the world. This is due to the great diversity of applications for which autonomous navigation can provide a better solution than previously existing methods. Most of these applications involve a need to carry out manipulations in areas where human presence is impossible, highly dangerous or otherwise impractical. An obvious example is the field of exploring the surface of other planets. NASA's Sojourner, the small explorer robot that was used by Mars Pathfinder in 1997, found traces of water on the surface of Mars, a fact of great importance which would have been practically impossible to learn without the use of an autonomous robot (see http://www.nasa.gov/). For the task of collecting samples from a coral reef under water, an autonomous robot may also be used efficiently, as it can stay under water for a long time, making it possible to collect a significant number of only probably interesting samples at a time [1]. Another application is a robot helicopter driver developed in Japan [2], which permits to put helicopters to various uses where the navigation task is less complex than usual, e.g. watering rice fields from helicopters without having to pay trained helicopter pilots.

There are also, however, more everyday situations, where the use of an autonomous robot can still be practical. An autonomous robot is able to perform several easy jobs which today take up valuable time of humans, for example office delivery, certain maintenance tasks, collecting litter in a fast food restaurant, and so on. The navigation process in these cases is quite similar: there is a known, well structured environment (an even floor, orthogonal walls, rectangular areas, doorways), which is also highly dynamic: humans – or other robots – may appear at any time. This makes it possible for us to consider the task of autonomous navigation within a building in general, regardless of the specific purpose for which the robot is used.

A vital part of the design of an autonomous robot is its navigation system, which has to decide at every given moment where to move next, taking in consideration all the a priori information on the environment, the sensory data and its knowledge of the current position and orientation as well as the goal position. The word "autonomous" is used here in the sense that the robot's navigation system is not intended to communicate with any remote system under normal circumstances.

Several different navigation methods have already been proposed, which can be categorized as global if the algorithm relies mostly on a priori information, or local if its decision is taken using mainly the current sensor data. Both global and local methods can be appropriate for several tasks, and have their own specific set of advantages and disadvantages; however, for our case, neither provides a solution alone.

The method of navigation proposed in this paper provides a way of blending the intelligence and optimality of global methods with the reactive dynamic behavior and low computational complexity of local ones. This is achieved by using a hybrid navigation system composed of two modules, one of which uses the a priori information and determines roughly the optimal route towards the goal, whereas the other carries out effective navigation decisions using the potential function based local approach, to which previous results have

In Proc. of the 12th IEEE Int. Conference on Fuzzy Systems, FUZZ-IEEE'2003, St. Louis, USA, May 25-28, 2003, pp. 266-271

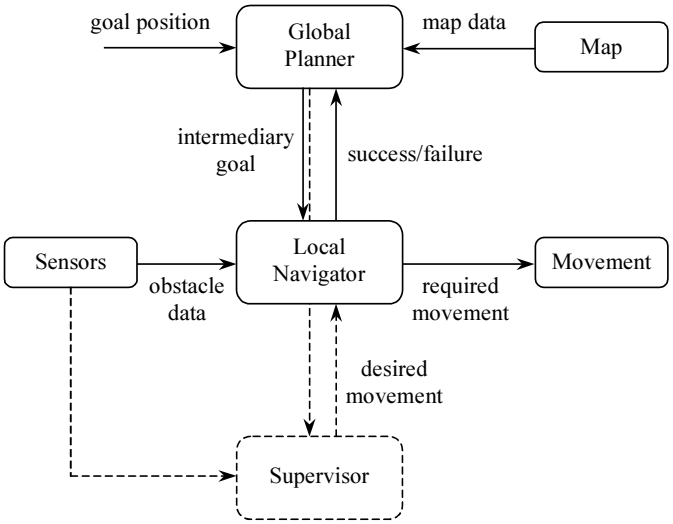


Figure 1. The main parts of the navigation system. Under normal circumstances, the supervisor block (below, dashed line) is not present, but it can be used in a possible teaching phase.

provided improvements such as complexity reduction, more flexibility, and a possibility to teach the system using a fuzzy representation. This latter part of the navigation system may also be used independently in an unknown part of the environment. The composition and the way of functioning of the complete system is shown on Figure 1.

The organization of the paper is as follows. Section 2 deals with the advantages and disadvantages of global and local navigation methods. Section 3 describes the global planning part of the navigation system, whereas in Section 4 the details of the proposed local navigation method are explained. Section 5 contains results of the simulation and Section 6 gives the conclusion.

## II. Global and Local Navigation

We call a navigation method global if its decision is taken using mainly a priori information available on the environment. A global algorithm generally yields a more intelligent result than a local one, as it has the ability to take into consideration every little detail of the navigation environment already known before the navigation task. There are global navigation algorithms – trajectory planning methods, generally based on previously existing optimization algorithms – that are able to find optimal trajectories according to various optimality criteria (see e.g. [3], [4]). The method described in [3], for example, works with two objectives: that the robot should pass as far from the obstacles as possible, and that the trajectory should be as short as possible – the weighing of these two is determined by a constant.

However, a global algorithm is unable to deal properly with dynamic situations. If a moving obstacle appears, the trajectory might have to be modified, i.e. recalculated, which in turn might take significantly more time than the movement of the obstacle. As a result, the robot might get "stunned" during the time it perceives the moving obstacle, always recalculating the trajectory to take, but finally moving on according to the original one. Another drawback of global algorithms is the need to have detailed knowledge of the environment. Perceiving an obstacle that has been moved into a previously free area can cause the same recalculation problem as a moving obstacle. The common cause of these disadvantages is the way global algorithms work, i.e. that they calculate the whole trajectory at the beginning of the navigation task, and whenever a new situation arises. This may result in a large quantity of resources being used up needlessly. Previous works dealing with global navigation methods all suggest the avoidance of such dynamic situations, none proposes a way of resolving them. Finally, a robot with a global navigation algorithm can of course only reach goal positions that are located on its a priori known map.

Local algorithms respond well to these challenges, as they decide the navigation course using current sensor information [5]-[7]. Most local navigation algorithms have the principal aim to navigate in a safe way, to avoid all the obstacles that appear during the navigation, while they usually continually approach the goal [8]. As a local algorithm never calculates parts of the trajectory in advance, its use of resources (computational power) is sensible if it is otherwise well designed. However, a local navigation algorithm will not guarantee an optimal trajectory by any optimality criterion. There may always be situations – local minima – where a local algorithm will even fail to find the target position: if, for example, the target and the robot are separated by a wall, the information gathered by the sensors is insufficient to find a way around the wall. Of course, local minimum situations do not necessarily exist; if obstacles are reasonably rare, a local navigation algorithm may still prove to be the best solution. This is also the case when we have no a priori information on the navigation environment [1], [9].

We can see that both local and global navigation methods suffer from major drawbacks, but none of these are in common. The best would be therefore to use a hybrid algorithm combining the advantages of the two approaches and eliminating their weaknesses. This idea is reinforced if we take a close look at the task of navigating the robot indoors. The facts that the environment is both known but also dynamic, thus unknown at times, impose the contradicting requirements of using both the a priori data and the momentary sensor data on the robot's navigation system. A global navigation method does not provide an appropriate solution here, as its behavior in the presence of dynamic obstacles is not satisfactory; whereas a local navigation method is very appropriate for navigating among dynamic obstacles, as it is very reactive, using always the newest information about obstacle positions, but it can easily get stuck into a local minimum position, where only an overall view of the environment would help to take the correct navigation decision.

The solution proposed in this paper resolves this

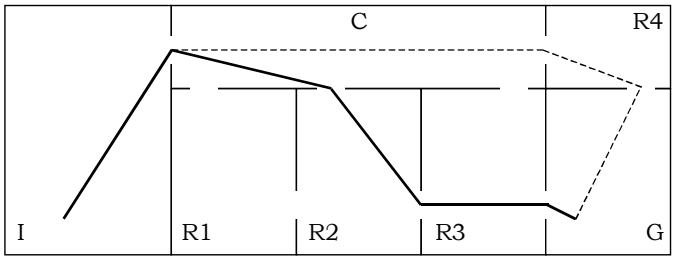


Figure 2. An example map. To get from an initial position in room I to a goal position in room G, the path that involves passing through the lowest number of regions (dashed line, according to Dijkstra's algorithm) is I-C-R4-G, whereas the shortest path found by the A* algorithm (solid line) passes through I-C-R2-R3-G.

contradiction by using a hybrid method which can blend the reactivity of local methods with the intelligence and optimality of global ones. The idea is to use global planning to determine only certain points – temporary goals – where we need to pass to reach the goal without getting stuck, then do the actual navigation by a local algorithm (see Figure 1). As mentioned in Section 1, there is always a risk that the local navigation method fails to reach a given target position. In most cases though, this risk is already greatly reduced by providing specific target positions that are always in the same region as the robot itself. It can also be supposed that in a real-world environment, the way from one door to another is not very complicated, as humans also use these passages (except for extreme situations such as when walls are being painted). Therefore there is a good chance that the robot will be able to reach the intermediary targets one by one.

This solution helps us preserve the reactivity and low computational complexity of local methods, while global information is also used to supervise the navigation – replan temporary goals if an unexpected situation occurs, a door is blocked, etc – and to eliminate local minimum situations by both avoiding and resolving them.

For the implementation of the global part of our proposed method, we introduce the application of the A* algorithm, which is widely known in other fields of artificial intelligence, and has proved to be advantageous or even optimal for similar problems. The local part is based on a fuzzy-neural realization of a modified version of the potential field based navigation technique, which offers a uniform, flexible, and adaptive solution to a wide range of navigation tasks. The following two sections present these two methods.

## III. Navigation in a Known Environment

To use a global planning for our navigation, we need to represent the information we know about the environment in which the navigation takes place, by storing a map of it. Several possible map representations have already been proposed and tested. We can use a gridmap of the environment [3], [10] – an occupancy grid, the graininess of which is still to be determined – or a topological map, which represents the environment with a graph, containing regions (rooms, corridors etc.) as vertices and passages (doors) as edges [11], [12]. In a dynamic environment it seems to be the best to use a topological map, as it is not possible to plan the details of the trajectory in advance, so the information stored in a relatively finely grained map would either remain unused or have to be reused each time a dynamic obstacle appears.

Using a topological map, we can determine at any moment what the possible ways to get to a certain goal are. However, if there are several different routes to the goal (due to cycles in the topological graph), it can be difficult to find all the possibilities, while we also need a method to find out which one is the most efficient. Choosing the route which involves passing through the smallest number of regions is, while sometimes certainly not the optimal solution (see e.g. Figure 2), a choice that can easily be implemented, using for example Dijkstra's well-known shortest path algorithm. According to measurements carried out by Thrun [13], the extra movement – thus also time – needed with this approach is not more than 3 per cent, but this figure seems to be highly dependent on the specific environment and the most common navigation tasks the robot is intended for.

The cause of this problem is the simple fact that different regions can have different shapes, thus passing through one of them can mean a much longer movement than passing through another. It would solve the problem if all regions were alike in shape and size, but unfortunately this is not possible, as dividing up larger regions into smaller ones would mean imposing unnecessary restrictions on the robot's movement. For example, moving along a long corridor would in this case require the robot to return to the middle of the corridor from time to time unnecessarily, to pass through an imaginary door. Also, passing through a room with a large number of doors may involve different movement lengths and times depending on which doors we use in the room – this makes this approach still a little imprecise.

Heuristically, the optimal route (i.e. the route that seems optimal before the navigation task) in an environment described by a topological map is the one which is the shortest, taking in consideration the distances from door to door in a straight line. We could also say that the optimal route is the one closest to the straight line. The number of all the possible routes from one point to another in the environment can be very high (it is practically doubled by each cycle in the topological graph), but fortunately we do not need to check all the possible routes to find the optimal one, we can use the mentioned heuristics.

The algorithm to use is the well-known A* (A-star) algorithm originally proposed also for solving typical shortest path problems (for a detailed description of the algorithm, see [14]). This algorithm finds the optimal route (the states to pass through) between an initial state and a goal state. The main idea in A* is to try to continue the route from the intermediary state which seems to be the most favorable, taking in consideration not only the cost of the way already made but

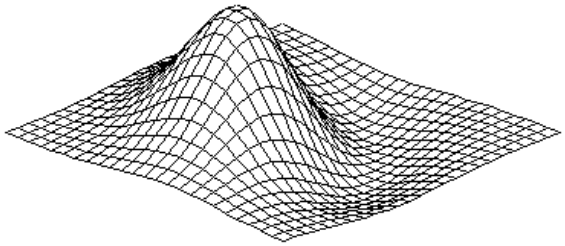

Figure 3. Potential function causing the robot to keep right. Obstacles to the left repulse the robot more than those to the right.

also the estimate on the cost of the remaining part of the solution. Thus, the algorithm needs an estimate on the remaining distance to the goal, which in our case is the distance along a straight line. It also needs a method to find all the states that are accessible from another in one step. In our case, this can be done by passing through the current region (room or corridor) to another door.

Like this, the A* algorithm is much more efficient than any other approach to the problem: it eliminates all the possible routes that involve making a detour, as the estimated remaining distance grows in this case as well as the distance already covered. The algorithm only examines states that are likely to be part of the optimal route, much alike to the way a human would do. The detailed description of the algorithm as used for this specific purpose can be found in [15].

With this algorithm, it is possible to find the optimal route to the goal. Of course, the route found may turn out not to be optimal if dynamic obstacles block or hinder the robot's passage while another way would be easier to pass through; the route is optimal in the sense that we can not find a better path using only the a priori information provided on the environment.

## IV. Navigation in a Dynamic Environment

After having planned the sections of the path to take using the A* algorithm, we need a local navigation method to carry out the effective movements of the robot required for reaching each intermediary goal (the door positions), while avoiding the obstacles – either static or dynamic – it encounters in a suitably reactive way. To achieve this, our robot needs to be equipped with a device that can provide sensory information on the immediate surroundings of the robot. If we take a look at previous works on autonomous navigation [3], [12], [16], we can see that two main types of sensors are suitable for this task: linear (ultrasonic or laser) sensors or cameras; in both cases, however, the pieces of information supplied to the navigation system are the positions of obstacles detected around the robot. Thus we can say that the decision of a local navigation system (or module) in the robot always depends on such obstacle position data.

Several different local navigation methods have been proposed, but there are two of these that seem to be general enough to be applicable under a variety of different circumstances (even in outdoor environments). One of these is the usage of – usually hierarchical – fuzzy rule bases [2], [17], by which so-called behavior based navigation methods can be efficiently formulated [7], [18], [19]. The other general local navigation method is the family of potential field based guiding methods [5], [8]. Within this issue, we have chosen an improved version of the latter because of its favorable complexity properties [20] and the possibility of realizing it using an adaptive and highly parallel fuzzy-neural implementation [20], [21].

The basic idea of potential field based navigation is that the navigation is safe if the robot moves around in a way as if all the obstacles repulsed it. So to determine the direction in which the robot should move, we imagine that every obstacle exerts a repulsive force on the robot, according to the expression

$$\underline{y} = -\sum_i p(d_i) \cdot \underline{e}_i \,, \tag{1}$$

where $d_i$ is the distance between the $i$th obstacle and the robot and $\underline{e}_i$ is the unit vector pointing towards the obstacle from the robot (i.e. the direction in which the obstacle is sensed). The function $p$ is the potential function, which takes its greatest values at the smallest values of $d_i$. The vector $\underline{y}$ is the obstacle avoidance vector; to determine the movement of the robot, it should be added to the vector pointing towards the goal position, with appropriate weighing constants that depend on the proximity of the nearest obstacle (if an obstacle is too close, we should concentrate first on avoiding it; if no obstacle is close, we can move towards the goal).

This first approach does not make it possible to distinguish between different directions, as we use the same function $p$ for every direction. In most cases though, we want the robot to behave differently when perceiving obstacles in different directions. The reason for this can be found either in the geometry of the robot itself, or in the general requirements we have for the navigation "style" of the robot [8]. To make such differentiation possible, we can generalize Expression 1 to have the following form:

$$\underline{y} = -\sum_i p(d_i \cdot \underline{e}_i) \cdot \underline{e}_i \,, \tag{2}$$

where the argument of the function $p$ is now a vector (the vector pointing to the obstacle from the robot) and the value of it is the magnitude of the imaginary repulsive force. In some cases, $p$ can take some negative values to specify attractive forces towards obstacles in certain positions.

This generalized form makes it possible to define different behaviors – modes, or styles – for the robot. For example, we can make the robot keep close to the obstacles while navigating, or keep right, by making the potential function asymmetrical. Figure 3 as an example shows the potential function which causes the robot to keep right.

Finally, another generalization that can be done to this

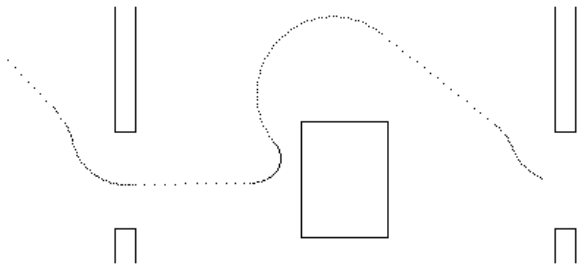

Figure 4. An extract of the navigation path of the simulated robot. A greater spacing between neighboring dots means that the robot moved with a greater speed.

approach is to use a vector function, one that directly gives the virtual force exerted by a detected obstacle in a given position:

$$\underline{y} = -\sum_i \underline{p}(d_i \cdot \underline{e}_i) \tag{3}$$

This vector function makes it possible to describe fields where an obstacle can not only repulse or attract the robot, but can also cause it to move in another direction, thus making it possible for example to make the robot alter its way when an obstacle appears far away in front of it. Like this, the navigation can become more smooth. Also, this generalization helps to eliminate the alternating behavior of scalar potential field based methods – i.e. that the robot tends to move in an oscillatory fashion when following a wall – described in [20].

The navigation method using such a vector function lends itself nicely to a fuzzy-neural implementation. The function itself can be represented by a universal fuzzy approximation described in [8]. Using this representation, the calculation of the navigation decision to take can be done by a fuzzy-neural structure that calculates the effect of each obstacle in a parallel way. This leads to a better performance regarding the time needed to evaluate the navigation situation.

The calculation complexity of this network can be further reduced using SVDR-based rule base reduction [20], [22], in which case the resulting neural structure consists of usually considerably less neurons per layer than before but needs to have a more general behavior than the standard MLP structure: nonlinearities are shifted from the neurons to the connections and each is approximated by a SISO fuzzy system. For details, refer to [23].

Another advantage of using the fuzzy-neural implementation mentioned above is the possibility to teach the local part of the system to navigate using a specific style, the decisions of which can be provided by any other system or even a human [21].

## V. Simulation Results

We tested the proposed hybrid navigation method by computer simulations. For this purpose, we have built an environment similar to Figure 2, which served both as a sample source of a priori information and an initial set of obstacles. In this environment, we placed additional objects (bricks) to verify whether the robot's navigation remains successful and safe in the presence of obstacles. At this stage of the simulation, dynamic (moving) obstacles were not implemented.

The navigation algorithm that was realized consisted of the two main parts described in the previous sections: the global planning module, which uses the A* algorithm, and the local navigation module using the potential field based method. The navigation itself is carried out in the following way. First, a goal position is specified by the user. Taking this goal position, the global planning module finds the optimal route to the goal and proposes the positions to pass through (the doors) as intermediary goals. These intermediary goals are then passed one by one to the local navigator, which makes the robot reach them while reactively avoiding the obstacles present in the environment, according to the potential function previously supplied to it.

To resolve possibly arising local minimum situations, the local navigation module was extended in a way that it checks periodically whether the robot has advanced towards the target lately, and if it detects that the robot is stuck, it asks the global navigator to replan another series of intermediary goals. This has, however, never happened in our simulation experiments. In Figure 4, an extract of the actual navigation path is shown (corresponding to room R3 in Figure 2). We can see that the robot managed to avoid the obstacle and reach the next intermediary goal.

This simulation was also intended for examining the properties of a future vision-based system. For this reason, a camera had to be simulated, a vision-based obstacle detection algorithm needed to be implemented and 3D images of the environment needed to be produced during the simulation. The obstacle detection algorithm that we used is based on the ideas described in [9] and [24]. It checks areas of the image of its environment against a pre-stored image of the floor both in RGB (Red-Green-Blue) and in HSV (Hue-Saturation-Value) representations, and it considers an area free if the difference between the two images is under a given limit.

The simulation was implemented in C/C++ on a PC running MS Windows 2000. Three-dimensional images were provided by standard OpenGL library calls. These choices were intended to support possible future porting, as both C/C++ and OpenGL are available on a wide range of platforms with minor differences in some basic conventions.

## VI. Conclusion

In this paper a navigation method has been proposed for autonomous robots used inside a building. The proposed method is a hybrid method composed of two main modules.

The first, global module uses the a priori information on the navigation environment and chooses some critical points to pass through before the actual navigation takes place. This module uses the A* algorithm to determine the optimal route to the specified goal position.

The second, local module then carries out the navigation itself, relying on current sensor data, thus making it easier to avoid static or dynamic obstacles. This module uses a fuzzy-neural representation of the potential field based navigation method.

The proposed method blends the intelligence and optimality of global navigation methods with the reactivity and low complexity of local ones. These statements are verified by a computer simulation of the complete system.

Possible further improvements include simulation in the presence of moving objects and testing in a real-world environment.

## REFERENCES


[1] Wettergreen, D., C. Gaskett, and A. Zelinsky, 1998, "Development of a Visually-Guided Autonomous Underwater Vehicle". In: *Proceedings of OCEANS98*. IEEE, 1998.

[2] Sugeno, M., H. Winston, I. Hirano, and S. Kotsu, 1996, "Intelligent Control of an Unmanned Helicopter Based on Fuzzy Logic". *NATO ASI on Soft Computing and Its Application,* Antalya, Turkey, August 1996.

[3] Blåsvær, H., P. Pirjanian, and H. I. Christensen, 1994, "AMOR – An Autonomous Mobile Robot Navigation System". In: *Proceedings IEEE International Conference on Systems, Man, and Cybernetics 1994,* vol. 3, pp. 2266-2271.

[4] Latombe, J., 1991, "Robot Motion Planning". Kluwer Academic Publishers, Boston, MA, USA.

[5] Borenstein, J., and Y. Koren, 1991, "The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots". In: *IEEE Transactions on Robotics and Automation,* vol. 7, n. 3, pp. 278-288.

[6] Saffiotti, A., 1997, "Fuzzy Logic in Autonomous Robotics: Behavior Coordination". In: *Proceedings of the 6th IEEE International Conference on Fuzzy Systems,* pp. 573-578, Barcelona, Spain, July 1997.

[7] Gaudiano, P., E. Zalama, and J. L. Coronado, 1996, "An Unsupervised Neural Network for Low-Level Control of a Wheeled Mobile Robot: Noise Resistance, Stability, and Hardware Implementation". In: *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, vol. 26, n. 3, pp. 485-496.

[8] Visontai, M., Sz. Szabó, P. Baranyi, A. R. Várkonyi-Kóczy, and L. Kiss, 2000a, "3-Dimensional Potential Based Guiding". In: *Proceedings of the IEEE Conference on INtelligent Engineering Systems (INES),* pp. 306-309. Portorož, Slovenia, September 2000.

[9] Lorigo, L. M., R. A. Brooks, and W. E. L. Grimson, 1997, "Visually-Guided Obstacle Avoidance in Unstructured Environments." In: *Proceedings IEEE Conference on Intelligent Robots and Systems,* Grenoble, France, September 1997.

[10] Gasós, J., and A. Saffiotti, 1999, "Integrating Fuzzy Geometric Maps and Topological Maps for Robot Navigation". In: *Proceedings of the 3rd Int. Symposium on Soft Computing (SOCO)*, Genova, Italy, June 1999.

[11] Thrun, S., J. Gutmann, D. Fox, W. Burgard, and B. Kuipers, 1998, "Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach". In: *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI),* pp. 989-995. Madison, WI, USA, July 1998. AAAI/MIT Press.

[12] Fabrizi, E., and A. Saffiotti, 2000a, "Extracting Topology-Based Maps From Gridmaps". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000)*, pp. 2972-2978, San Francisco, CA, USA, April 2000.

[13] Thrun, S., 1998, "Learning Maps for Indoor Mobile Robot Navigation". In: *Artificial Intelligence,* vol. 99, pp. 21-71, 1998. Elsevier Science.

[14] Russell, S. J. and P. Norvig: "Artificial Intelligence – A Modern Approach". Prentice-Hall, 1995.

[15] Kiss, L., A. R. Várkonyi-Kóczy, "A Hybrid Autonomous Robot Navigation Method Based on Artificial Intelligence and Soft Computing Techniques". Submitted to the IFAC International Conference on Intelligent Control Systems and Signal Processing (ICONS'03), Faro, Portugal, April 2003.

[16] Cheng, G., and A. Zelinsky, 1998, "Goal-oriented Behaviour-based Visual Navigation". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'98)*, Leuven, Belgium, May 1998.

[17] Saffiotti, A., E. H. Ruspini, and K. Konolige, 1999, "Using Fuzzy Logic for Mobile Robot Control". In: Practical applications of fuzzy technologies. H.-J. Zimmermann, ed, pp. 185-206. Kluwer Academic Publishers, 1999.

[18] Soliday, S. W., 1995, "A Subsystem Approach to Developing a Behavioral Based Hybrid Navigation System For Autonomous Vehicles". Master's Thesis, Department of Electrical Engineering, North Carolina A&T University, Greensboro, NC, USA, 1995. URL: http://www.soliday.com/stephen/research/thesis/index.html

[19] Fabrizi, E., and A. Saffiotti, 2000b, "Behavioral Navigation on Topology-Based Maps". In: *Proceedings of the 8th International Symposium on Robotics with Applications*, Maui, Hawaii, USA, June 2000.

[20] Visontai, M., Sz. Szabó, A. R. Várkonyi-Kóczy, P. Baranyi, G. Samu, and L. Kiss, 2000b, "Complexity Problem of the Potential Based Guiding". In: *Proceedings of the IFAC Symposium on Artificial Intelligence and Real-Time Control (AIRTC),* pp. 145-149. Budapest, Hungary, October 2000.

[21] Korondi, P., A. R. Várkonyi-Kóczy, Sz. Kovács, P. Baranyi, M. Sugiyama, "Virtual Training of Potential Function Based Guiding Styles". Joint 9th IFSA World Congress and 20th NAFIPS International Conference, IFSA/NAFIPS 2001, July 25-28, 2001, pp. 2529-2534.

[22] Yam, Y., P. Baranyi, C. T. Yang, 1999, "Reduction of Fuzzy Rule Base via Singular Value Decomposition". *IEEE Transactions on Fuzzy Systems*, vol. 7., pp. 120-132.

[23] Baranyi, P., Y. Yam, H. Hashimoto, P. Korondi, and P. Michelberger, "Approximation and Complexity Reduction of the Generalized Neural Network". Accepted to *IEEE Transactions on Fuzzy Systems*.

[24] Gaskett, C., L. Fletcher, and A. Zelinsky, 2000, "Reinforcement Learning for a Vision Based Mobile Robot". Robotic Systems Laboratory, The Australian National University, 2000.