A Universal Vision-based Navigation System for Autonomous Indoor Robots

László Kiss, Annamária R. Várkonyi-Kóczy

Integrated Intelligent Systems Japanese-Hungarian Laboratory
Department of Measurement and Information Systems
Budapest University of Technology and Economics

Magyar tudósok körútja 2., H-1117 Budapest, Hungary

1 Introduction

The topic of autonomous robotics is very popular nowadays because of the great diversity of applications for which an autonomous robot can provide a better solution than previously existing methods. Initially, the main concern of researchers of this topic was to make manipulations possible in areas where human presence is impossible, highly dangerous or otherwise impractical. An example of such an application is NASA's Sojourner, the small explorer robot that was used by Mars Pathfinder in 1997, which found traces of water on the surface of Mars, a fact of great scientific importance (see http://mars.jpl.nasa.gov/MPF/). Another such task is the collection of samples from a coral reef – an autonomous robot 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]. It is easy to imagine several more such situations, like manipulations within a nuclear reactor, or even driving a helicopter using an autonomous robot [2], which could permit to put helicopters to various uses where the navigation task is less complex than usual without having to pay trained helicopter pilots.

However, the use of an autonomous robot can still be practical in some much more everyday situations. Several easy jobs which today take up valuable time of humans could be performed by a robot, for example certain maintenance tasks, collecting litter, office delivery, and so on. In these cases, the navigation process is quite similar: there is a (more or less) 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 allows us to consider the task of autonomous navigation within a building in general, regardless of the specific purpose for which the robot is used.

This work was sponsored by the Hungarian Fund for Scientific Research (OTKA T 035190)

A vital part of the design of an autonomous robot is its navigation system (in a general sense), which has to decide at every given moment where to move next, taking into 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. There are three main questions to answer in this design process: what kind of sensors should be used; how to use the sensor-provided information for the purposes of the navigation; and how to use the a priori information.

To all of these questions, several answers have been proposed, each adapted to a certain specific type of navigation task. In this paper, we present a general solution to navigation indoors, which combines the advantages of previous approaches and results in an intelligent, adaptable navigation system.

The organisation of the paper is as follows. Section 2 describes the problems we encounter when trying to design a universal navigation system. Section 3 deals with the possible ways of obstacle detection and explains the method proposed. Section 4 presents the local part of the navigation system, which uses sensor information, whereas in Section 5 the global part which uses the a priori information is explained in detail. In Section 6 some results of the simulation of the complete system are shown and Section 7 gives the conclusion.

2 A Universal Navigation System

A navigation system for an autonomous robot has the main task of avoiding every obstacle, while reaching the specified goal position. As mentioned above, there are two main types of information that can be used for this purpose (apart from knowing the robot's current position and orientation): previously supplied, a priori information on the navigation environment in general, and sensor-provided data describing the immediate surroundings of the robot at the moment of sensing. These two types of information obviously require to be dealt with in a different way. So far, navigation systems have always been based on mainly the a priori – global – information, or mainly the sensor-provided – local – data, therefore the algorithms they use are also categorised as global or local.

The use of global algorithms generally leads to a more intelligent navigation behaviour than that of local ones, as global methods have the ability to take all the details of the navigation environment into consideration, if they are known a priori. Some global navigation algorithms – mainly trajectory planning methods that are based on previously existing optimisation algorithms – are able to find an optimal trajectory according to various optimality criteria [3] [4]. However, global

algorithms have their drawbacks; the most important among these is probably their inability to deal properly with dynamic situations. On the appearance of a moving obstacle, the trajectory may have to be modified, i.e. recalculated, which in turn might take significantly more time than the movement of the obstacle itself. As a result, the robot will either slow down or even stop during the time it perceives the moving obstacle, or use a lot more resources than needed. It is also a disadvantage of global methods that to be able to use them, the navigation environment must be known precisely in detail. If an obstacle gets moved into an area that is marked as free, it can lead to the same recalculation problem as a moving obstacle. Finally, a robot with a global navigation algorithm can of course not navigate in regions outside its a priori known map.

All these disadvantages are due to the fact that global algorithms calculate the entire 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, or an inappropriate navigation behaviour.

Local algorithms take their navigation decisions according to current sensor information, thus they respond well to dynamic challenges, and may be used in an unknown environment as well [5] [6]. A local navigation algorithm is usually safe, it is able to avoid all the obstacles that appear during the navigation, while it continually approaches the goal. As a local algorithm calculates only the movement to carry out in the next moment, its use of resources (computational capacity) is sensible if it is otherwise well designed. However, local algorithms have none of the nice intelligence and optimality properties of global methods. An optimal trajectory – by any optimality criterion – can not be guaranteed with a local navigation algorithm. Moreover, the possibility of the existence of local minimum situations – situations where a given local algorithm will fail to find the target position – can not be eliminated: if, for example, the target and the robot are separated by a wall, the information gathered by the sensors may be insufficient to find a way around the wall. Of course, the existence of local minimum situations is not necessary but only a threat; a local navigation algorithm may still prove to be the best solution if obstacles are reasonably rare or we have no a priori information on the navigation environment [1].

It can be seen that both local and global navigation algorithms have major disadvantages, but none of these are in common. In the following, we propose a hybrid navigation method, which can combine the intelligence of global navigation algorithms with the reactive dynamic behaviour of local ones. The proposed method uses global planning and a priori information to specify intermediary goals for the navigation, then does the actual navigation using a local algorithm. This approach helps us preserve the advantages of both classes of navigation algorithms: as the effective navigation decisions are carried out by a local module, a suitable dynamic behaviour can be reached, while local minimum situations can be on the one hand avoided by choosing appropriate – locally reachable – intermediary goals, and on the other hand resolved by detecting that

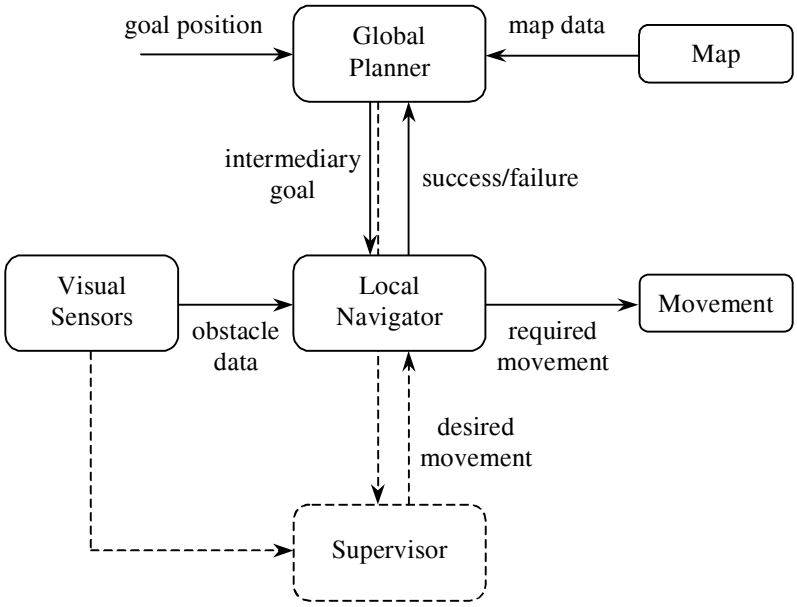

Figure 1. The main parts of the navigation system. Under normal circumstances, the supervisor block (below, dashed line) is not present, but it can be used in a possible teaching phase.

the robot got stuck and planning another series of intermediary goals. The components and the way of functioning of the complete system is shown on Figure 1.

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 realisation 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. In the following sections, the components of this system are presented.

3 Detecting Obstacles

To detect and avoid static and dynamic (moving) obstacles around the robot, it must be equipped with a device that can provide sensory information on its immediate surroundings. Taking a look at previous works on autonomous

navigation [3] [7] [8], we can see that two main types of sensors are suitable for this task: linear (ultrasonic or laser) sensors or cameras, or even both, as they both have advantages over each other.

The pieces of information supplied by the sensors to the navigation system are in all cases the positions of obstacles detected around the robot. Thus the usage of linear sensors is quite comfortable, as they directly provide the position of the first obstacle sensed in the corresponding direction, whereas with a camera, we need an algorithm which "notices" the obstacles on the images seen, and calculates their position. However, if we use linear sensors only, some obstacles may remain unnoticed if they can be sensed in a direction in which none of the sensors is pointing, which cannot happen if we use a camera. Also, a robot without a camera is not general enough for all types of indoor tasks, as detecting the objects to be manipulated probably needs a camera anyway. Therefore we have decided to use vision for the task of detecting obstacles.

There are three main approaches to the problem of obstacle detection based on vision: stereo vision based detection, optical flow based detection and obstacle detection based on monocular image features. The next two subsections present these methods.

3.1 Vision-based Obstacle Detection: Using Multiple Images

Stereo vision based obstacle detection [9] has the main idea of capturing two images of the environment at the same time, thus using a similar system of vision as that of humans. The positions of obstacles can then be determined in two ways. One solution is to calculate the precise location of the points in each image via triangulation; this requires, however, a way to determine which pixels in the two images correspond to each other, which is, apart from being algorithmically difficult, a task that has inconveniently high computational power requirements.

Another way of determining the obstacle positions is provided by the so-called inverse perspective mapping [10], which maps the pixels of the two images to the ground plane, as if they all represented points of the ground; then the obstacle positions can be calculated using the difference of the two mapped images, since they are going to differ only if obstacles are present. However, this approach also needs a lot of computational capacity, while it suffers from the problems described in [9] as well. We can thus conclude that stereo vision based detection is not suitable for our purposes.

Obstacle detection via optical flow [11] [12] takes its main idea also from nature: the vision of insects. (The term "optical flow" refers to the movement of the pixels of the image seen.) Based on the empirical fact that objects closer to a viewer seem to move quicker when the viewer is moving, it calculates the distance of the robot from each pixel in the image. This approach also uses more than one image

to determine where obstacles are around the robot, but it needs only one camera and uses the images captured in different moments. Its main drawback, similarly to stereo vision, is the computation complexity caused by the need to find which pixels in two images correspond to each other.

3.2 Vision-based Obstacle Detection: Using a Single Image

The third approach does not share the disadvantage of high computational capacity requirement, as it uses only a single image of the environment. A little surprisingly, this can be enough to detect the obstacles even in an unstructured environment, also highly homogeneous in colour [13], by simply using both the RGB (Red-Green-Blue) and HSV (Hue-Saturation-Value) representation of pixels.

The method described in [13] was designed for exploration on the surface of Mars. If the environment has parts that are not known in advance, this is an obstacle detection method that fits our purposes. However, if we have sufficient a priori information on the whole of the environment, and we are able to store the image of the ground in advance, we can use another method comparing sections of the image of the robot's surroundings against such pre-stored ground images [14], which works indoors without major mistakes even if only the RGB representation is used.

A possible improvement of this latter method is to use both the RGB and HSV pixel representations. Thus the steps of the detection algorithm are the following: we partition the image of the environment into squares of 8 by 8 pixels, calculate the mean RGB and HSV of these pixels and check them against the pre-stored mean RGB and HSV of the floor. If the difference exceeds a certain limit, we consider that the 64-pixel image segment came from an obstacle, the position of which can be determined from the direction in which we see it; otherwise the image contains part of the floor, so we haven't detected an obstacle.

4 Avoiding Obstacles: The Potential Field

Having a method of detecting obstacles around the robot, our next concern is to move the robot towards the target, while also avoiding the obstacles detected. This task will be performed by the local part of the system described in Section 2, which uses the information provided by the obstacle detection algorithm.

Several different local navigation algorithms have already 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] [15], by which so-called behaviour based navigation methods can be formulated efficiently [16] [17]. The

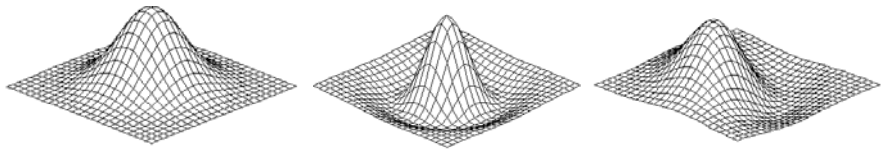

Figure 2. Three potential functions describing different navigation styles: keep as far from obstacles as possible (left), close to obstacles (centre) or close to those on the right (right).

other general local navigation method is the family of potential field based guiding algorithms [5] [18]. For our experiments with the hybrid navigation method here proposed, we have decided to use the latter because of its advantageous complexity properties and its highly parallel, adaptable fuzzy-neural implementation [19] [20]. Below only the main idea of this navigation method is described; for details of complexity reduction and fuzzy-neural implementation, refer to [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

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

where d_i is the distance between the ith 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), thus the argument of the function p is the vector pointing to the obstacle from the robot. The function p is the potential function; its value is the magnitude of the imaginary repulsive force exerted by an obstacle perceived in the given relative position $d_i \cdot \underline{e}_i$. To make the navigation safe, p should take large values at small 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). In some cases, p can take some negative values to specify attractive forces towards obstacles in certain positions.

A major advantage of this method is that it makes it possible to define several different behaviours – modes or styles – for the navigation of the robot. For example, we can specify a function p that causes the robot to navigate as far from each obstacle as possible; or to keep close to obstacles; or to keep a greater distance from obstacles on the left than from those on the right. Figure 2 as an example shows the potential functions describing these three movement styles.

This first approach usually provides safe navigation, and if obstacles are placed rarely enough, it also finds its way to the target position. However, it needs improvement for two reasons. First, if the robot has to move along a long wall, it will move in an oscillatory fashion [19]. Second, it does not always take the most intelligent decision using the available sensor data: if an obstacle is far away to the front, the best would be to shift the robot sideways, but this would only be possible if the obstacle could exert a force in another direction, not only in a central one. Both of these weaknesses can be eliminated using the following generalised approach:

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

Here the function $\underline{p}$ has a vector value – it gives directly the virtual force exerted by the obstacle. Of course, to make sure that the navigation is still safe, we need to choose a function $\underline{p}$ which takes large and centrally directed values at small values of d_i, like before.

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 [18]. 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 or computational power needed to evaluate the navigation situation.

The calculation complexity of this network can be further reduced using SVD (Singular Value Decomposition) based rule base reduction [19] [22], in which case the resulting neural structure consists of usually considerably less neurons per layer than before, though it has a more general structure than the standard Multi-Layer Perceptron: 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 this local part of the system exactly what its navigation decisions should be, where the desired decisions can be provided by any other system or even a human [20].

5 Reaching Targets: The A* Algorithm

Using the algorithm presented in the previous section, the robot is capable of avoiding the static and dynamic obstacles it encounters. If the robot had to move around in an unknown environment, this would be the most intelligent navigation algorithm to provide the robot with: if the building has unknown parts, the robot

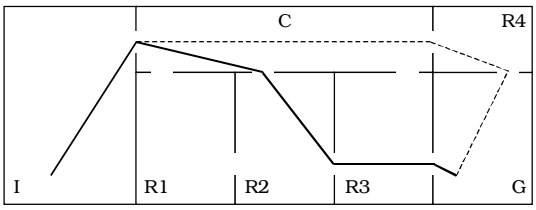

Figure 3. An example map. Dijkstra's algorithm suggests the route I-C-R4-G, while the optimal route found by A* is I-C-R2-R3-G.

may navigate using only the presented local navigation algorithm. However, in most of the cases, we have extensive knowledge of the topology of the building in which the navigation takes place, which enables us to use one more algorithm in order to make the navigation of the robot more intelligent.

As mentioned in the previous section, local algorithms usually find the target positions if obstacles are reasonably rare. This is, however, not the case with buildings: walls are obstacles, and they appear quite often. To solve such situations, we need a method which can make use of the a priori information on the environment. This method is the one called "Global Planner" in Figure 1.

First the form of the a priori information must be decided, i.e., what kind of map we are going to use. Several possible map representations have already been proposed and tested. We can use a gridmap of the environment [3] [24] – 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 [7] [25]. In a dynamic environment like most buildings, 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, not to mention that local decisions are taken intelligently enough by the local navigation algorithm already.

If we use a topological map, the intermediary goal positions can only be the positions of the doors we want the robot to pass through. It seems that supplying door positions as intermediary targets is a choice by which the navigation can be carried out, because the potential field based approach reaches the target in most cases if obstacles between the current position and the goal position are placed rarely enough, which is usually the case with doors in the same region.

Using a topological map, we can determine at any moment the possible ways to get to a certain goal. 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 3), a choice that can easily be implemented, using for example Dijkstra's well-known shortest path algorithm [26]. Fortunately, we can use another algorithm that finds the optimal route quickly enough.

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 into 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 [27]). 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 favourable, taking into consideration not only the cost of the way already made but 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. Notice also that the first result found by the A* algorithm is always the best solution, as the estimated remaining distance is always less than or equal to the actual remaining distance. The detailed description of the algorithm as used for this specific purpose can be found in [28].

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.

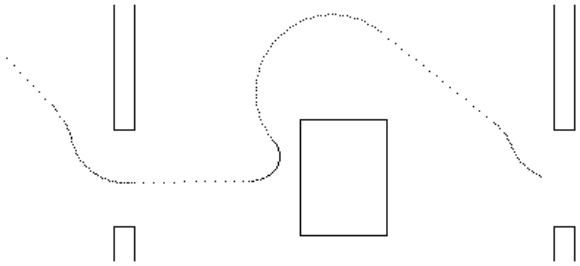

Figure 4. An extract of the navigation path of the simulated robot. A greater spacing between neighbouring dots means that the robot moved with a greater speed.

6 Simulation Results

We tested the proposed navigation system by computer simulations. For this purpose, we created a virtual environment in which the robot could move around, which could be displayed in 3 dimensions from arbitrary points of view.

The topology of this sample virtual environment was the one shown by Figure 3, in which we have tried to represent every topological situation that a robot in a real environment may encounter. In addition to the walls, we placed some objects (bricks) in random places 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 realised consisted of the two main parts described in sections 4 and 5: 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. The detection of the obstacles is done using the algorithm described by section 3.2; the 64-pixel image sections captured by the simulated camera are produced by small 3D views of the environment.

To resolve possibly arising local minimum situations (situations where the local navigation algorithm fails to proceed), 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 3). We can see that the robot managed to avoid the obstacle and reach the next intermediary goal.

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.

7 Conclusion

In this paper a general navigation system has been proposed for autonomous robots used inside a building. The proposed method is composed of three main parts.

The first part of the system is a vision-based obstacle detection module, which detects both static and dynamic (moving) obstacles around the robot. This part uses two representations of pixel colours, and checks segments of the captured image against that of the floor.

The second part is a local navigation module that carries out the navigation itself, relying on the current sensor data, thus avoiding static or dynamic obstacles. This module uses a fuzzy-neural implementation of the potential field based navigation method.

The third part is a global navigation module, which 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 proposed method blends the intelligence and optimality of global navigation methods with the reactivity and low complexity of local ones. It also eliminates the disadvantages of obstacle detection based on linear sensors. 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, "Development of a Visually-Guided Autonomous Underwater Vehicle". *Proceedings of OCEANS98.* IEEE, 1998.

[2] Sugeno, M., H. Winston, I. Hirano, and S. Kotsu, "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, "AMOR - An Autonomous Mobile Robot Navigation System". *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics,* vol. 3, pp. 2266-2271, 1994.

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

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

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

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

[8] Cheng, G., and A. Zelinsky, "Goal-oriented Behaviour-based Visual Navigation". *Proc. of IEEE Int. Conf. on Robotics and Automation,* Leuven, Belgium, May 1998.

[9] Bertozzi, M., A. Broggi, and A. Fascioli, 1996, "Real-Time Obstacle Detection using Stereo Vision". *Proceedings EUSIPCO-96 – VIII European Signal Processing Conference*, pp. 1463-1466, Trieste, Italy, September 1996.

[10] Mallot, H. A., H. H. Bulthoff, J. J. Little, and S. Bohrer, 1991, "Inverse Perspective Mapping Simplifies Optical Flow Computation and Obstacle Detection". *Biological Cybernetics,* vol. 64, n. 3, pp. 177-185.

[11] Grünewald, M., and J. Sitte, 2001, "A Resource-Efficient Approach to Obstacle Avoidance via Optical Flow". *Proceedings of the 5th International Heinz Nixdorf Symposium: Autonomous Minirobots for Research and Edutainment (AMIRE),* vol. 97 of HNI-Verlagsschriftenreihe, pp. 205-214. Heinz Nixdorf Institute, October 2001.

[12] Steinkraus, K., and L. P. Kaelbling, 2001, "Optical Flow for Obstacle Detection in Mobile Robots". Artificial Intelligence Laboratory, MIT, 2001.

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

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

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

[16] Soliday, S. W., 1995, "A Subsystem Approach to Developing a Behavioral Based Hybrid Navigation System For Autonomous Vehicles". Master's Thesis, Dept. of Electrical Engineering, North Carolina A&T University, Greensboro, NC, USA, 1995.

[17] Fabrizi, E., and A. Saffiotti, "Behavioral Navigation on Topology-Based Maps". *Proceedings of the 8th Int. Symp. on Robotics with Applications,* June 2000.

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

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

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

[21] Kiss, L., A. R. Várkonyi-Kóczy, and P. Baranyi, 2003, "Autonomous Navigation in a Known Dynamic Environment". Submitted to: IEEE Int. Conf. on Fuzzy Systems, May 2003, St. Louis, MO, USA.

[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] Gasós, J., and A. Saffiotti, "Integrating Fuzzy Geometric Maps and Topological Maps for Robot Navigation". *Proc. of the 3rd Int. Symp. on Soft Computing*, June 1999.

[25] Thrun, S., J. Gutmann, D. Fox, W. Burgard, and B. Kuipers, "Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach". *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI),* pp. 989-995, July 1998.

[26] Dijkstra, E. W., "A Note On Two Problems In Connexion With Graphs". Numerische Mathematik, n. 1, pp. 269-271, 1959.

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

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