A PATH PLANNING METHOD FOR AUTONOMOUS MOBILE ROBOTS

Ognyan B. Manolov
e-mail: omanolov@bas.bg
Laboratory for Autonomous Mobile Robots, Institute of Control and System Research, Bulgarian Academy of Sciences
P. O. Box 79, 1113 Sofia, Bulgaria

Key words: Environment model, data structure, visibility graph, path planning.

ABSTRACT

Mobile robots often find themselves in a situation where they must find a trajectory to another position in their environment, subject to constraints posed by obstacles and the capabilities of the robot itself. This is the problem of planning a path through a continuous domain, for which several approaches have been developed. A method for autonomous mobile robot path planning is presented. Initially, the environment model, given as a closed chain of polygonal obstacles, is transformed into a visibility graph of obstacle vertices with a minimum number of links. An additional visibility graph of obstacle is formed simultaneously. The given initial point and destination point are presented as obstacles with a single vertex and are added to the determined graphs as vertices and as obstacles correspondingly. The extended graphs are updated and in the first step a shortest path from initial point to destination point through the obstacles is searching for. By this method a subset of obstacles is selected. On this basis from the visibility graph of obstacle vertices a sub-graph is detached, containing only the vertices of obstacles, belonging to the selected subset. We search the detached sub-graph to find the shortest path between the given points. For determination of the visible obstacle vertices a fuzzy logic algorithm is implemented

I. INTRODUCTION

The path-planning problem is as old as mobile robots and is a fundamental problem to the adaptation of artificial intelligence technology to robotics, but is not one that has found a universal solution. A single definition of the "path-planning problem" or the "obstacle-avoidance problem" does not exist. There is an ensemble of techniques that assume varying degrees of knowledge about the external world [1, 2, and 5]. This family of problems might be summarised as the generation and execution of action in order to move the mobile robot to the desired position and orientation without violating certain physical constraints. One of the relatively recently developed tools that may help tackle the problem of real-time path planning are Rapidly-exploring random trees (RRTs) [3]. RRTs employ randomization to explore large state spaces efficiently, and can form the basis for a probabilistically complete though non-optimal kinodynamic path planner [4]. Their strengths are that they can efficiently find plans in high dimensional spaces because they avoid the state explosion that discretizations faces. Most current robot systems that have been developed to date are controlled by heuristic or potential field methods at the lowest level, and many extend this upward to the level of path navigation [5]. Since the time to respond must be bounded, reactive methods are used to build constant or bounded time heuristics for making progress toward the goal. One set of reactive methods that have proved quite popular are potential fields and motor schemas [6]. Although they meet the need for action under time constraints, these methods suffer from the lack of look ahead, which can lead to highly non-optimal paths and problems with oscillation [7].

The path planning in a 2D world with polygonal obstacles has received considerable attention as a part of the general problem of mobile robot motion planning, which has a number of competitive solutions. More of the known methods involve using graph search algorithms to find the minimum distance path in the graph of the possible paths. Generally, the graph of possible paths is a "visibility graph", or ***V-Graph*** of the obstacle vertices, where an arc connects two vertices if and only if they are visible from one another. A vertex **V** is defined as being visible from a vertex **W** if the segment **VW** does not intersect any obstacle edges in the planning space [8, 14].

An algorithm for solving the shortest-path problem is known, which is based on Dijkstra's algorithm, [9], and uses a full visibility graph in $\mathbf{O}(\mathbf{n}^2)$ time, where **n** is the number of obstacle vertices. In [10] a "Sector theory" is given, where in a result, at most two vertices per obstacle being included in the search graph. Several facts as local tangency and "overlapping obstacle profiles" are used in [11, 15], and as it is noted in [12], they can be applied substantially to reduce the search graph.
In [12], the problem of path planning, in the sense given above, is considered as a problem of the level of a

hierarchy of control. An algorithm, which requires **O (n^2)** time for generated visible vertices to given vertex, is presented in [13]. The path planning algorithm integrates some pruning rules of the search graph size reducing, with a hierarchical set of tests for a candidate vertex visibility.

Here a path planning method is presented, which is based on a two-steps graph search. Initially the given environment model is transforming as a set of polygonal obstacles into a visibility graph of obstacle vertices. It will be shown that such a graph with a minimum number of links exists there, which provides the minimum Euclidean distance path between the given initial and destination points. The visibility graph of obstacles will be determined simultaneously. Preparing the initial data ends with determination of these two graphs. The given initial and destination points are presented as polygons with single vertex and are added to the determined graphs as a vertex and as an obstacle, correspondingly. The extended graphs are updated and in the first step a rough shortest path from initial point to destination point through the obstacles is searched for. By this method a subset of obstacles is selected. It will be claimed the shortest path from the initial point to the destination point in this environment to pass through the vertices of this subset only. On this basis, from the visibility graph of obstacle vertices a sub-graph is detached, containing only the vertices of obstacles belonging to the selected subset. The detached sub-graph is searching to find the shortest path between the given points. Finally, the process of planned path execution, which is based on the fuzzy approach, described in [18], is implemented.

II. AN ENVIRONMENT MODEL

The environment model as a closed chain of polygonal obstacles is given. Every obstacle is presented by a closed chain of its vertices. Since there is no principal difference between the obstacles and there is no need to assign a number to each of them – obstacle1, obstacle2,… The same is valid for obstacle vertices except that they are ordered clockwise or counter clockwise.

II.1. The Data Structure of the obstacles

The structure of the obstacle is formed by a closed chain of structures from Vertex type. There is no first vertex since they are "equal in rights". That is why there are no numbers in the corresponding cells, presented in Figure 1. The first element of structure Vertex is k-triple of the vertex parameters, including its Cartesian coordinates and the type of the vertex that means the vertex can be temporary or permanent, in the sense given in [12]. In such a way this structure can be use in case of operating in an unknown or partially known environment. The closed chain of vertices is formed by the next two elements in the structure: a pointer to the next vertex and a pointer to the previous vertex. To ensure the independence from the direction of passing through the vertices two pointers are used. It is useful to have the passing opportunity toward both directions in the future data processing. The next pointer in this structure, pointer to corresponding node, provides an interaction between the environment-model data structure and the ***V-Graph*** structure of the obstacle vertices. The last element in the structure Vertex is a marker that is used for the data processing.

The environment is presented as a closed chain of structures from type **Obstacle** as it is shown in Figure 1. The structure **Obstacle** is analogous to the structure **Vertex**, but its elements have different function.

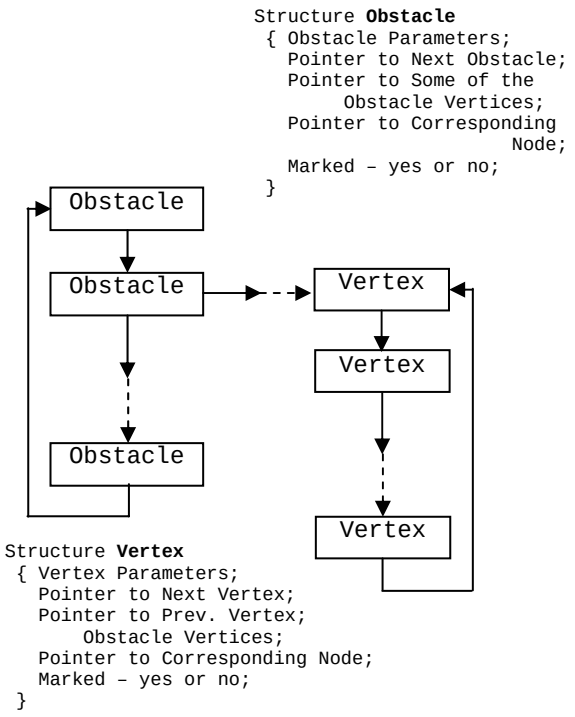


Figure 1. The Environment-Model Data Structure

The obstacle's parameters involve its barycentre Cartesian coordinates and its type, which indicates if the obstacle is "closed" or "open", when it is partially or completely inspected. The next element is the pointer to the following obstacle in the chain. Since the obstacle vertices are "equal in rights" it does not matter which vertex will be directed to the next pointer. The next pointer in the Structure **Obstacle** is the pointer to corresponding Node. It provides the interaction with the environment-model data structure and the structure of the ***V-Graph*** of obstacles. The marker has the same function as in the structure **Vertex**.

II.2. The *V-Graph* Data Structure

There are many ways of graph representation. We propose the representation given in Fig. 2., where two main structures are used: **Node** and **V-Node**. The Structure **Node** forms the chain of graph nodes. The Structure **V-Node** is used to represent the nodes that are connected

with a given node. The Structure **Node** consists of the following elements:
- the pointer to some of the corresponding V-nodes (it does not matter to which of the nodes connected to the treated node this pointer is directed);
- the pointer to the next node in the graph (to form the chain);
- and the pointer directed to the node data structure, either to the structure Obstacle or Vertex, if we deal with the obstacle ***V-Graph*** or the vertex ***V-Graph,*** respectively .

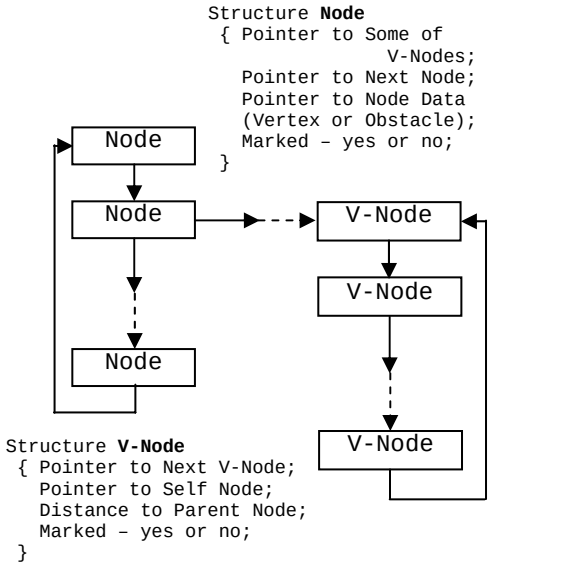


Figure 2. The ***V-Graph*** Data Structure

The Structure **V-Node** forms the chain of nodes that are connected with a given node. It consists of a pointer to the next node in the chain of the nodes that are connected to the given, the pointer to the self node in the chain of the Structure **Node**, the Euclidean distance to the given node, and a marker.

II.3. Obstacles Descriptions

Consider the relations between two obstacles $\mathbf{O_i}$ and $\mathbf{O_j}$ as they are shown in Figure 3. There are four common tangents between them in the condition that obstacles are convex. The method used to determine the "upper" and the "lower" tangents is given in [16].

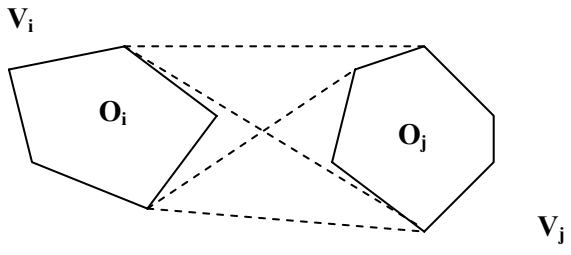


Figure 3. The relations between two obstacles.

Let $\mathbf{V_i}$ be a vertex of the obstacle $\mathbf{O_i}$ and $\mathbf{V_j}$ be a vertex of the obstacle $\mathbf{O_j}$. Let there be no vertex of $\mathbf{O_j}$ visible from $\mathbf{V_i}$ and no vertex of $\mathbf{O_i}$ visible from $\mathbf{V_j}$. There are four ways of going around both obstacles. Therefore the shortest path between $\mathbf{V_i}$ and $\mathbf{V_j}$ will contain one of the four common tangents. The found endpoints of the common tangents are checked for visibility by using of a coordinate transformation only in accordance with [12] but modified, as it will be shown below. The vertex ***V-Graph*** with the minimum number of links is the graph with nodes in case of the obstacle vertices and two nodes are connected with an arc, if and only if:

1) they are visible and the connecting segment is a common tangent to the corresponding obstacles, or

2) they are connected with an obstacle edge.

The obstacle ***V-Graph*** is defined as a graph with nodes in case of the obstacle vertices and two obstacles are connected with an arc if and only if there exists at least one vertex from the first obstacle and at least one vertex from the second obstacle, such that they are connected with an arc of the vertex ***V-Graph***. The algorithm for determination of the vertex ***V-Graph*** with the minimum number of links, [17], and the obstacle ***V-Graph*** simultaneously determination is formulated as follows:

1. *Create a basic vertex **V-Graph** with nodes all vertices and with arcs the corresponding obstacle edges;*
2. *Partition all non-convex obstacles into convex ones;*
3. *Sort the obstacles in ascending $\boldsymbol{X}_{min}$ order;*
4. *For each pair of obstacles:*
   a. Determine the "interesting" obstacles:
   b. Determine the common tangents;
   c. If the corresponding vertices are visible
   *{ update vertex **V-Graph**;*
   *if obstacle **V-Graph** is not updated*
   *(update **V-Graph**); }*

It has to note here some of the main algorithm's features. First of all, a hierarchical test to reduce the search space is used. The processing of non-convex obstacles is difficult, especially when their convex hulls are intersected, [16]. By reason of this, the second feature is that every non-convex obstacle is partitioned into convex one. This approach is illustrated in Figure 4., which shows the principle of partitioning the non-convex polygon ABCDE into convex polygons AFE and BCDF.

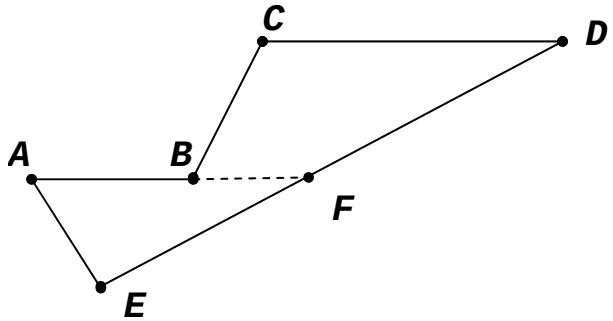


Figure 4. Partitioning of the obstacle

More detailed description of this method is given in [17]. Further it will be supposed that all obstacles are convex. The obstacles are sorted in ascending $\mathbf{X}_{\min}$ order using standard Quick Sort procedure in time **O** (log **n)**, where **n**

is the total number of the vertices of both obstacles. With this step the following data processing became easier.
An obstacle is defined to be "interesting" if the following conditions are fulfilled:

$$\mathbf{X}_{min} \le \boldsymbol{Right}\ , \qquad \mathbf{X}_{max} \ge \boldsymbol{Left}\ ,$$

$$\mathbf{Y}_{min} \le \boldsymbol{Up}\ , \qquad \mathbf{Y}_{max} \ge \boldsymbol{Down}\ ,$$

Where ***Right, Left, Up*** and ***Down*** are the Cartesian coordinates **X** and **Y** of the boundaries corresponding to the enveloping rectangle of the treated obstacles, as it is shown in Figure 5.

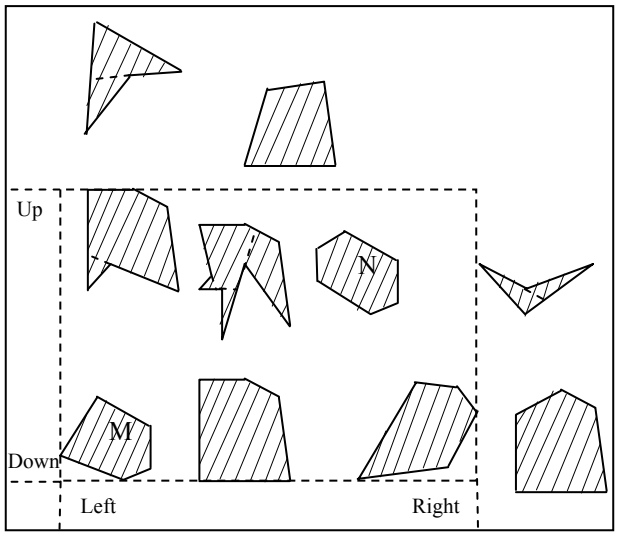


Figure 5. The determination of the vertex **V-Graph**

III. PATH PLANNING METHOD

It is trivial to determine the path when the destination point is visible from the initial one. In the other hand, the given initial point (**IP**) and destination point (**DP**) both are presented as polygons with a single vertex. The possible successors are determined in an analogical way to the method presented in [12] and are illustrated in Figure 6.

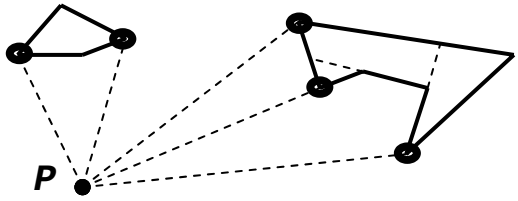


Figure 6. The determination of possible successors.

It can be seen that only the vertices, which satisfy the explicit conditions, described below, are picked out. For example, the segment connecting the given point **P** and the treated vertex should be tangent to the obstacle. The found successors are marked with ⊙ in Figure 6. The proposed path planning method is realized as algorithm, described as follows:

*1. If **DP** is visible from **IP** then exit with path to **DP**.*
*2. Determine possible successors from **IP** and from **DP**.*
*3. Update vertex **V-Graph** and obstacle **V-Graph** by adding **IP** and **DP**, with the arcs to possible successors, like a vertex and like an obstacle correspondingly.*
*3. Search obstacle **V-Graph** for shortest path.*
*4. Determine "interesting" obstacles and form a sub-graph of the vertex **V-Graph** that contains only vertices of the "interesting" obstacles.*
*5. Search the determined sub-graph for shortest path from **IP** to **DP**.*

The **IP** and **DP** with their successors are added to the determined vertex ***V-Graph*** and obstacle ***V-Graph*** as a vertex and as an obstacle correspondingly. The extended graphs are updated and in the first step we search for a rough shortest path from **IP** to the **DP** through the obstacles. We use a modified version of Dijkstra's algorithm involving the triangle rule. By this method we select a subset of obstacles (including **IP** and **DP)**.
The determination of the "interesting" obstacle in the described above path planning algorithm is analogous to the approach for determination of the ***V-Graph***. But here the bounds of the corresponding rectangles are formed for every two adjacent obstacles in the selected subset of obstacles, as it is shown in Figure 7.

The obstacles found to be "interesting" are added to the selected subset. The next step is to detach from the vertex ***V-Graph*** a sub-graph, which contains only these vertices that belong to the obstacles in the selected subset. Further, we search the determined sub-graph to find the minimum-distance path using a modified version of Dijkstra's algorithm.

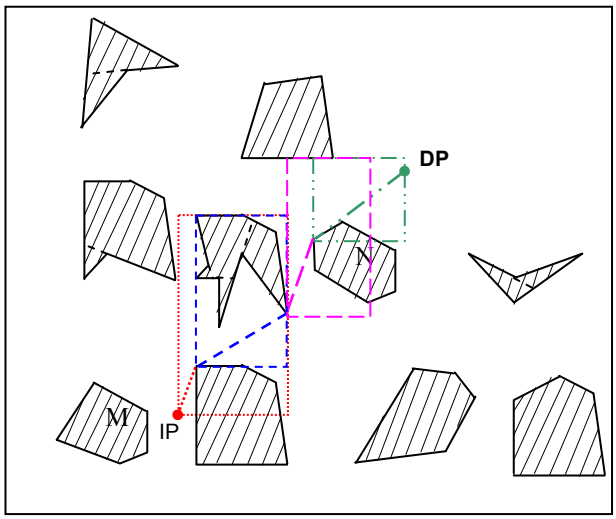
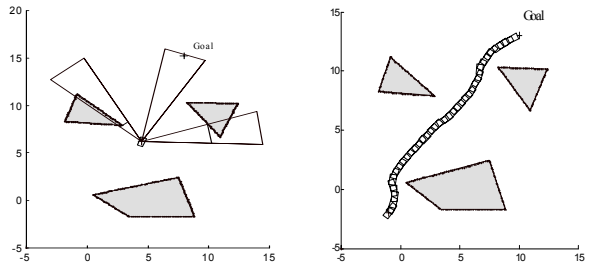


Figure 7. The path planning method

IV. AN EXECUTION OF THE PLANNED PATH

A fuzzy approach to path control of an autonomous mobile robot is implemented, which provides robot movement round the obstacle on a safe distance, until the obstacle falls out of the path towards the goal [18]. The control strategies are modelled and represented by if-*then*

fuzzy rules. The mobile robot has received the mission to reach a given goal position from given start position through the shortest path, which has been obtained by the path planning method, presented above. The robot is equipped with scanning ultrasonic rangefinder placed on its platform. The sensor is capable to measure in a set of symmetric fixed positions relatively to the robot's direction, as it is shown in Figure 8. a). The initial heading of the robot was directed to the goal. A fixed environment was considered. The both technics, path planning method and fuzzy approach to path execution control of autonomous mobile robot have been experimented by computer simulation experiments on standard PC using the *Fuzzy Logic Toolbox* of *Matlab*. One of the results is presented in Figure 8.b).



a) b)

Figure 8. Simulation experiments of the robot movement

V. CONCLUSION

The purpose of this work was to provide a suitable and handy data structure to meet the requirements of path planning in an environment with polygonal obstacles. The representation of the search space as a visibility graph of obstacle vertices with the minimum number of links reduces considerably the number of alternatives in searching for the shortest path. Using a two-step graph search, first in obstacle **V-Graph** and second in a sub-graph of vertex **V-Graph** with the minimum number of links, greatly reduces the search space. The presented method is applicable to path planning in a known environment with polygonal obstacles but the proposed data structure can be used for operating in an unknown environment. The method for path planning in unknown or partially known environment and the integration of sensor information with the proposed data structure is a significant step to the autonomous mobile robots navigation. It will be the purpose for further investigations and realizations.

ACKNOWLEDGMENT

This result was obtained in the frame of ICSR Project, founded by BAS, 2003. The author is much obliged to Prof. M. Isabel Ribeiro from the IST/ISR, Technical University of Lisbon, and to Prof. M. M. Konstantinov with the Sofia University of Architecture and Civil Engineering, for their useful discussions and shared experience in the field of mobile robotics.

REFERENCES

1. Cox, I. J., G. T. Wilfong (ed.), *Autonomous Robot Vehicles*, Springer-Verlag, 1990.
2. Tzaffestas, S. G., Fuzzy and Neural Approaches to Robot Control, *Int. Conf. on Advanced Robotics and Intelligent Automation*, Athens, 1995, pp. 34-54.
3. S. M. LaValle, Rapidly-exploring random trees: A new tool for path planning. *Technical Report No. 98-11*, October 1998.
4. S. M. LaValle and J. James, J. Kuffner, Randomized kinodynamic planning. *Int. Journal of Robotics Research, Vol. 20, No. 5*, p.p. 378–400, May, 2001.
5. J.-C. Latombe, *Robot Motion Planning.* Kluwer, 1991.
6. R. C. Arkin, Motor schema-based mobile robot navigation. *Int. Journal of Robotics Research, August 1989, 8(4)*, p.p. 92–112, 1989.
7. J. Bruce, M. Veloso, Real-time Randomized Path Planning for Robot Navigation. *Technical Report No.3-8*, Grant No DABT63-99-1-0013, CSD - CMU, Pittsburg, PA, USA, 2000.
8. Lizano-Perez T., M.A. Wesley, An algorithm for Planning Collision-free Paths among Polyhedral Obstacles. *Communications of the ACM,* Vol. 22 (10), 1979.
9. Asano T., L. Guibas, J. Hershberger, H. Imai, Visibility of Disjoint Polygons. *Algorithmica,* Vol. 1 (1), 1986.
10. Koch E., Planning the Robot Motion in Binary Space. *Master's thesis, (reprint)*, Gainesville, FL, 1984.
11. Mitchell J., Planning Shortest Pats. *Ph.D. Thesis,* Stanford University, 1986.
12. Montgomery M., D. Gaw, A. Meystel, Navigation Algorithm for a Nested Hierarchical System of Robot Path Planning Among Polyhedral Obstacles. *Proc. of IEEE Int. Conf. on Robotics & Automation*, Raleigh, North Carolina, 1987.
13. Meystel A., Knowledge-based Controller for Intelligent Mobile Robots. *Master's thesis, (reprint)*, Drexel University, Philadelphia, PA, 1986.
14. Lizano-Perez T., Automatic Planning of Manipulator Transfer Movements. *IEEE Trans. on SMC*, Vol. SMC-11, 1981.
15. Narshing Deo, Chi-yin Pang, Shortest-Path Algorithms: Taxonomy and Annotation. *Networks*, Vol. 14, 1984.
16. Aggarwal A., Computational Geometry - Lecture notes. Massachusetts Institute of Technology, 1988.
17. Raitchev R., E. Neitchev, Transformation of the Environment Model into V-Graph with Minimum Number of Links. *Problems of Engineering Cybernetics & Robotics*, Vol. 35, Bulg. Academy of Sciences, 1989.
18. Stanev P., E. Enchev, Sv. Noikov, O. Manolov, J. Zaprianov, Fuzzy Control and Reflexive Behaviour of an Autonomous Vehicle in an Unknown Environment. *Proc. of 3rd IFAC Symposium on "Intelligent Autonomous Vehicles",* March 25-27, 1998, Madrid, Spain.