Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 96 (2014) 59 – 69

Procedia Engineering

www.elsevier.com/locate/procedia

Modelling of Mechanical and Mechatronic Systems MMaMS 2014

Path planning with modified A star algorithm for a mobile robot

František Duchoň*[a], Andrej Babinec[a], Martin Kajan[a], Peter Beňo[a], Martin Florek[a], Tomáš Fico[a], Ladislav Jurišica[a]

[a]*First affiliation, Address, City and Postcode, Country*
[b]*Second affiliation, Address, City and Postcode, Country*

Abstract

This article deals with path planning of a mobile robot based on a grid map. Essential assumption for path planning is a mobile robot with functional and reliable reactive navigation and SLAM. Therefore, such issues are not addressed in this article. The main body of the article introduces several modifications (Basic Theta*, Phi*) and improvements (RSR, JPS) of A star algorithm. These modifications are focused primarily on computational time and the path optimality. Individual modifications were evaluated in several scenarios, which varied in the complexity of environment. On the basis of these evaluations, it is possible to choose path planning method suitable for individual scenario.

Peer-review under responsibility of organizing committee of the Modelling of Mechanical and Mechatronic Systems MMaMS 2014

Keywords: path planning, A* algorithm, Basic Theta*, Phi*, Jump Point Search

Nomenclature

SLAM	Simultaneous Localization and Mapping
A*	A star
RSR	Rectangular Symmetry Reduction
JPS	Jump Point Search

* Corresponding author. Tel.: +421 915 719 462
E-mail address: frantisek.duchon@stuba.sk

Peer-review under responsibility of organizing committee of the Modelling of Mechanical and Mechatronic Systems MMaMS 2014
doi:10.1016/j.proeng.2014.12.098

1. Introduction

The term path planning was developed in many fields, such as robotics, artificial intelligence or control theory. That is why each scientist uses own definition of this term [1]. In robotics, path planning concerns with problem as how to move a robot from one point to another point. With the advances in robotics path planning also includes many complications such as uncertainties [2], multiple robots [3-5], or dynamics [6]. In artificial intelligence, path planning means a search for a sequence of logical actions that transform an initial robot state into a desired goal state. Such planning may include many decision-theoretic ideas such as Markov decision processes [7], imperfect state information [1], learning methods [8,11] or game-theoretic equilibrium [9]. In the control theory, path planning deals with issues of stability [10], feedback [11], and optimality [12-13]. As it can be seen, path planning of a mobile robot is a wide problem and there exist many methods and approaches to it.

Generally in robotics, path planning is focused on designing algorithms that generate useful motions by processing simple or more complicated geometric models [1]. This article is focused on such algorithms. Path planning in robotics can be divided in three main groups [1] – motion planning, trajectory planning and planning under uncertainty. Path planning addresses the automation of mechanical systems that have sensors, actuators, and computation capabilities. In [1] the motion planning and trajectory planning are defined as a fundamental needs in robotics that are described by algorithms that convert high-level specification of task from humans into low-level description of how to move. This is described by a Piano Mover's Problem. The task is defined with a precise model of house and a piano as input to an algorithm. The algorithm must determine how to move the piano from one room to another without hitting anything. Robot's path planning is defined in similar way. However, motion planning usually ignores dynamics and other constraints and focuses primarily on the translations and rotations of controlled object – robot. Recent research in this area considers also other aspects such as uncertainties [14], differential constraints [15], optimality [12-13] etc. Trajectory planning usually refers to the problem of taking the solution from a motion planning and determining how to move along this solution with regards to mechanical limitations of the robot [16-17].

A common task for a mobile robot is to navigate in an indoor environment [1]. A robot might be asked to perform tasks as building a map of the environment [18], determining its precise location within a map [19], or arriving at a particular place without collision [20] etc. First two tasks are usually connected into one problem called SLAM (Simultaneous Localization And Mapping) [19, 21-25]. This issue is a well-known problem and nowadays there are a lot of open source solutions addressing this problem – e.g. HectorSLAM [24], OpenSLAM [25], etc. Navigation of the robot in an environment is dependent on the knowledge of the environment. If the robot does not have any description of environment, it uses its sensors to "feel" this environment. On the basis of this data, it can navigate from one place to another. Such approach is called reactive navigation, because robot only reacts on the stimulus from sensors [26]. However, if the robot has a precise map of the environment than it can uses many techniques to navigate in optimal way. This is called path planning or global navigation. In this article, the problem of path planning is described and several methods of path planning are evaluated. The basic assumptions for evaluated methods are:

- functional SLAM,
- no dynamic constraints of the robot's movement,
- path is generated as the connection between the points of interest (usually where robot changes its orientation).

Every path planning method is dependent on a state space. State space represents all possible positions and orientations of a robot. There are several ways how to describe the state space. Usually stat space is represented implicitly by a planning algorithm. That is why the path planning algorithms can be divided on the basis of used state space. The mostly used representation of state space in robotics is grid/metric map. Although geometric or topological representations are used, the metric map is easy to update new information about the environment. However, one cell from the grid represents only small amount of space. That is why metric map may have a large size. The nature of most currently used sensors in robotics results in their frequent use. Algorithms evaluated in this paper are also based on a grid representation of space.

All planning problems involve a sequence of decisions that is applied over time. Moreover, in the planning formulation, it must be specified how the state changes when actions are applied. In mobile robotics, it is usually a state-valued function formed as a sequence of states from configuration space that the robot must reach. Each path planning also involves initial and goal state. There are generally two different kinds of planning concerns based on the type of criterion. First is feasibility – find a plan that causes arrival of the robot at a goal state, regardless of its efficiency. Second is optimality – find a feasible plan that optimizes performance in some specified manner [1]. Algorithms in this paper are only feasible. However, some optimal criterions were defined to compare the performance of these algorithms:

- computational time,
- length of path,
- number of examined cells,
- symmetry of examined environment.

The last criterion was added, because each of the evaluated algorithms handles with this property of the environment in a different way. The memory requirements for all of the algorithms are almost the same, because they are most dependent on environment representation, which was still the same. That is why this criterion was not included.

2. A* algorithm

A* algorithm is one of the best known path planning algorithm, which can be applied on metric or topological configuration space [27]. This algorithm uses combination of heuristic searching and searching based on the shortest path. A* algorithm is defined as best-first algorithm, because each cell in the configuration space is evaluated by the value:

$$f(v) = h(v) + g(v) \tag{1}$$

Where $h(v)$ is heuristic distance (Manhattan, Euclidean or Chebyshev) of the cell to the goal state and $g(v)$ is the length of the path from the initial state to the goal state through the selected sequence of cells. Obviously, this sequence ends in the actually evaluated cell. Each adjacent cell of actually reached cell is evaluated by the value $f(v)$. The cell with the lowest value of $f(v)$ is chosen as the next one in the sequence. Advantage of this algorithm is that the distances used as a criterion can be adopted, modified or another distance can be added. This gives a wide range of modifications of this basic principle. For example time, energy consumption or safety can be also included in function $f(v)$. A* algorithm, as a path planning algorithm for a mobile robot, was successfully implemented and tested. Results can be found in [28]. However, these results were not entirely satisfactory and good. The computational time was very large. For the map containing 60,000 cells, the computation on average computer lasted over one hour. This is quite discouraging for the application of this algorithm to the robot. Therefore, further investigation was conducted to the possibilities of modification of the A* algorithm.

3. Modifications of A* algorithm

Basic A* algorithm used for a grid configuration space is restricted to 8-connectivity (Fig. 1). This means that it can find path that is based on connection between the closest possible cells. However, this is not quite useful, because there can be a lot of free space between the connected cells over long distances and these cells may not be linked by zigzag style. That is why the searching in every angle is introduced (Fig. 1).

Fig. 1. Restriction of 8-connectivity searching in A* algorithm (left) and the searching in every angle (right).

Algorithms which use the searching in every angle are Basic Theta* and Phi*. Basic Theta* [29] is an extension of A* algorithm, which resides in the test of the visibility between cells. This means that if the tested cell has a direct visibility to the cell included in selected sequence, the cells between them are ignored. In this way only cells, which robot has to pass, are found. These cells are characterized also with the change of the robot's orientation (Fig. 2).

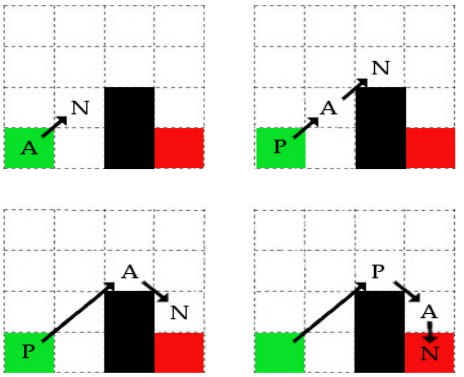

Fig. 2. Basic Theta* algorithm. Green cell is a initial state, red cell is a goal state, black cells represent the obstacles. In the second step, the cell N has direct visibility to the cell P. That is why the cell A is ignored and the cell P and N are directly connected.

Phi* algorithm [30] is an extension of the Basic Theta* algorithm. The extension resides in the recording of a local predecessor of each evaluated cell. Logically, this predecessor is one of the neighborhood (by the definition of Basic Theta*) cells. This algorithm also records two angles for each cell. These angles define the range in which the predecessor can be found (Fig. 3). This property of Phi* algorithm allows in some restrictive way to introduce the dynamics of the robot into the calculation of the algorithm.

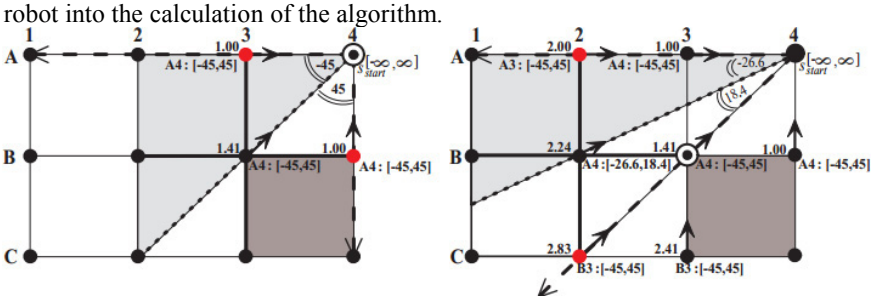

Fig. 3. Phi* algorithm. Each cell has a range of angles in which the predecessor of the cell can be found.

However, with enhanced capabilities of these algorithms it is not possible to plan the path in a real time. These algorithms still uses complex grid maps. That is why a Rectangular Symmetry Reduction (RSR) was implemented. RSR is proposed as a preprocessing step, which identifies and eliminates the symmetries. Advantages of RSR are low computational demands and possibility of combination with the family of star algorithms [31]. The symmetries are defined as the rectangles inside which only free space occurs. Path inside these rectangles is defined only by the set of the rectangle's edges. If the initial, actual or goal state is inside the rectangle, cell representing this state is temporary added as a new cell. This temporary cell is connected with all peripheral cells of the correspondent rectangle (Fig. 4). By this data reduction, the computational time of path planning algorithms can be reduced ten to hundred times.

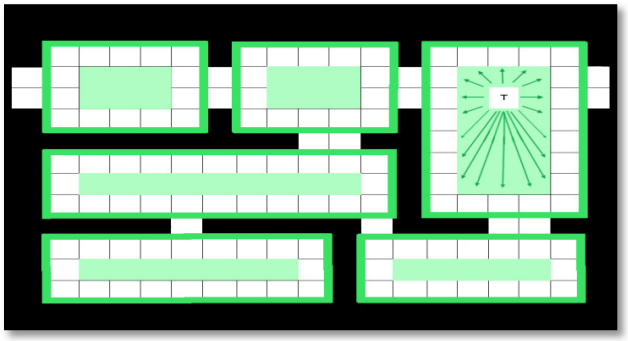

Fig. 4. Temporal addition of the cell T with corresponding edges. The number of cells used in path planning algorithm has been reduced from 175 to 20.

Another method how to reduce the amount of the examined cells is Jump Point Search (JPS) [32], which can be directly applied on A* algorithm. The principle of this method is the cropping of neighborhood cells in the surroundings of actually evaluated cell. The principle can be demonstrated on simple example in Fig. 5. Let the a denotes the actually evaluated cell and the arrow marks denote the movement from the previous cell. Then the grey unoccupied cells can be cropped, because they can be reached from the previous cell without the need of entering the cell a. The remaining cells (white color) are called natural neighbors. In ideal case, only natural neighbors are taken into account for path planning. The example in Fig. 5 has no obstacle cell. Obstacles in JPS create the forced neighbors (Fig. 6). These neighbors are created, because some cells cannot be cropped due to the blockage of alternative paths to the cells.

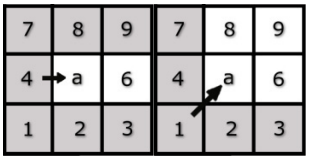

Fig. 5. Cropping of neighborhood cells in the surroundings of cell a (unoccupied cells).

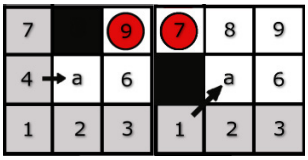

Fig. 6. Cropping of neighborhood cells in the surroundings of cell a (with one obstacle cell). Red circle denotes the forced neighbor.

These principles can be demonstrated in Fig. 7. JPS algorithm starts from initial state a. It examines recursively to the right, until it reaches cell b. This cell fulfilled the rule with a forced neighbor. To reach the forced neighbor c, it is needed to jump from the cell a to the cell b. Cells between a and b are not taken into account. The number of evaluated cells is significantly reduced by this manner. For an examplne in Fig. 8 (120 cells), the number of evaluated cells was only 8.

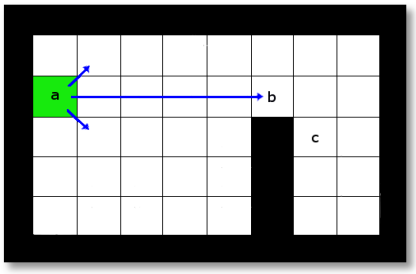

Fig. 7. Example of JPS method.

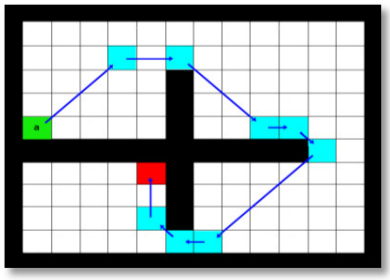

Fig. 7. Path planning with JPS method. The initial state is green, the goal state is red, blue cells denote the jump points.

4. Experiments

For the verification of described algorithms, standard office environment was used. The map of the environment (Fig. 8) was created by a robot on the basis of laser rangefinder and selected SLAM technique. RSR was applied on the map. The result of this reduction can be seen in Fig. 9. These are the basic assumptions for the experiments. Each algorithm was verified on several scenarios varying in symmetry of environment. Computational time, number of examined cells and length of path were evaluated. For the proper verification of the algorithms especially in terms of computational time, 500 repetitions of algorithm's calculation were performed. The average time from all repetitions was calculated. In this manner, variable computational power in OS Windows was reflected.

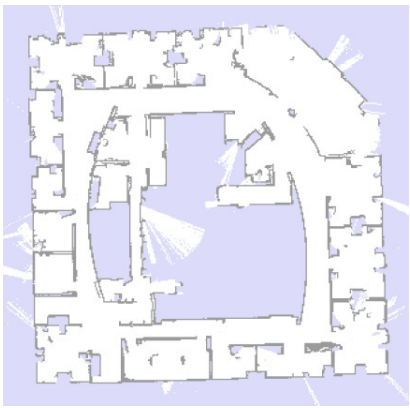

Fig. 8. Map of the environment used for experiments.

Fig. 9. Application of RSR on the map.

First of all, the algorithms were tested if they are able to find the path between the initial and goal state. The output of each algorithm is created as a set of points, in which robot changes its direction. However, for better clarity these points are connected into a continuous path. It may be stated that all of the algorithms are able to find path between the initial and goal state. As it can be seen from results (Fig. 10), each algorithm searched a different amount of space (yellow points). It is clear that evaluated algorithms have different advantages and disadvantages. This is summarized in Table 1. For the precise evaluation, other scenarios were performed.

Table 1. Summary of the characteristics of evaluated algorithms

Algorithm	Advantages	Disadvantages
A*	Simplicity	Cannot search in every angle
	Relatively fast	In basic form does not uses RSR
	Modifiability	
Basic Theta*	Search in every angle	Long calculation time
	Resultant path is created as a set of points	
	Finds shorter paths than 8-connectivity algorithms	
Phi*	Search in every angle	Long calculation time
	Resultant path is created as a set of points	
	Finds shorter paths than 8-connectivity algorithms	
	Fast replanning	
JPS (A*)	Fast	Cannot search in every angle
	Good reduction of symmetries	
	Resultant path is created as a set of points	

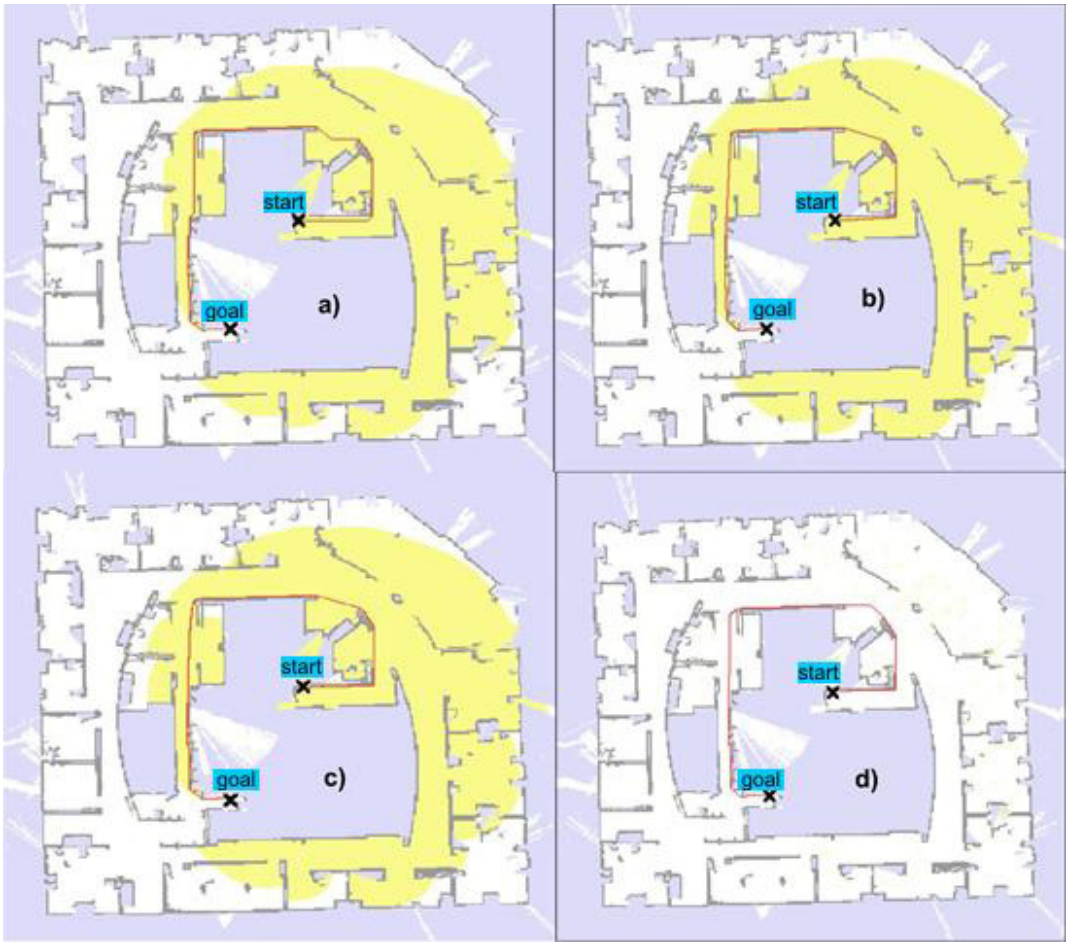

Fig. 10. First scenario (find path between initial and goal state): a) A* b) Basic Theta* c) Phi* d) JPS (A*).

Second scenario (Fig. 11) was characterized by environment with high symmetry, i. e. initial and goal state were placed inside of such area. In that area there are many paths with the same length, therefore it was important to compare these algorithms in this scenario. Results can be seen in Table 2. From the results it is clear that the fastest algorithm is JPS (A*). This algorithm also examined the smallest amount of space. However, determined path by this algorithm is not the shortest. The shortest path was found by algorithms (Basic Theta* and Phi*), which are able to examine the path in every angle. However, their computational time is 50 times longer than JPS (A*).

Table 2. Summary of the second scenario.

Algorithm	Computational time [ms]	Length of path [cells]	Examined cells [cells]
A*	229	499,3625	26736
Basic Theta*	998	477,3128	29599
Phi*	1019	477,5218	29019
JPS (A*)	21	499,3625	757

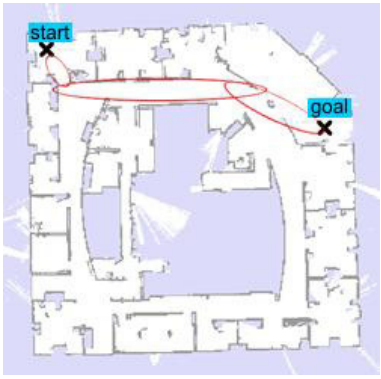

Fig. 11. Second scenario: areas with high symmetry are marked with red ellipse.

Third scenario (Fig. 12) was characterized by environment with average symmetry. As it can be seen in Fig. 12 the size of symmetry areas is reduced. Also the path in this case is about 20 percent longer. Results can be seen in Table 3. From the results it is clear that the fastest algorithm is JPS (A*) again. This algorithm also examined the smallest amount of space. However, determined path by this algorithm is longer than the one calculated by standard A* algorithm. The shortest path was found by Basic Theta* and Phi* algorithms again. However, their computational time is almost 100 times longer than JPS (A*). This is caused by lower symmetry of examined space than in the previous scenario.

Table 3. Summary of the third scenario.

Algorithm	Computational time [ms]	Length of path [cells]	Examined cells [cells]
A*	353	601,8011	57655
Basic Theta*	2347	580,0478	61431
Phi*	2898	582,6714	62076
JPS (A*)	30	605,3158	1161

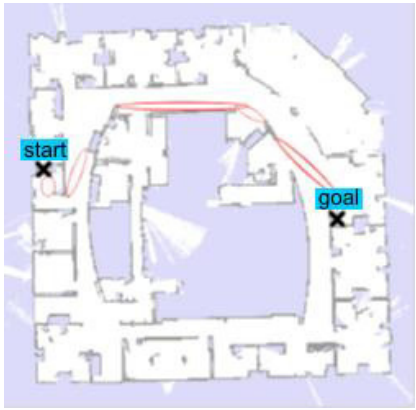

Fig. 12. Third scenario: areas with average symmetry are marked with red ellipse.

Fourth scenario (Fig. 13) was characterized by environment with low symmetry. Identification of such environment is very complicated, because used map contains many areas with high symmetry. Results can be seen in Table 4. From the results it is clear that the fastest algorithm is still JPS (A*). This algorithm also examined the smallest amount of space. However, the amount of examined cells is the same for A*, Basic Theta* and Phi*. This is caused by low symmetry of examined space. Algorithms, which use search in every angle, are unable to prosper from this feature. This time it can be said that the shortest path was found by Basic Theta* (2.28% shorter than the longest one).

Table 4. Summary of the fourth scenario.

Algorithm	Computational time [ms]	Length of path [cells]	Examined cells [cells]
A*	447	595,2670	69240
Basic Theta*	3518	581,9875	76368
Phi*	4473	584,1820	75712
JPS (A*)	34	595,2670	1248

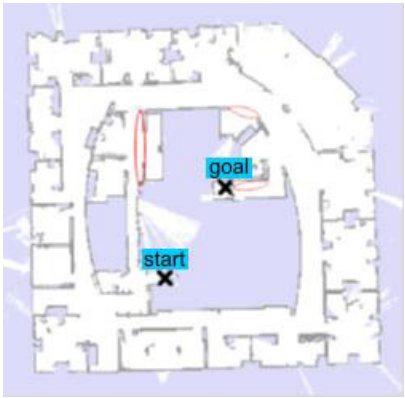

Fig. 13. Fourth scenario: areas with low symmetry are marked with red ellipse.

5. Conclusion

In terms of fast finding of the path, JPS (A*) was the best algorithm in various environments. Its disadvantage is that it found longer path, often longer than the standard 8-connectivity algorithms, e. g. A* algorithm. Therefore, this algorithm is suitable to use in cases where it is necessary to quickly find a path. If the computational time is not significant and length of path is especially important then it is suitable to use Basic Theta* algorithm. This ability of Basic Theta* has been highlighted especially in environments with lower symmetry.

Acknowledgements

This work has been supported by projects VEGA 1/0177/11, APVV-0539-11, VEGA 1/0178/13 and KEGA 003STU-4/2014.

References

[1] LaValle, S. M., 2006. Planning Algorithms, Cambridge University Press, New York, USA.

[2] Van Den Berg, J., Abbeel, P., Goldberg, K., 2011. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. The International Journal of Robotics Research, 30(7), p. 895-913.
[3] Liu, S., Sun, D., Zhu, C, 2011. Coordinated motion planning for multiple mobile robots along designed paths with formation requirement. Mechatronics, IEEE/ASME Transactions on, 16(6), p. 1021-1031.
[4] Svetlík, J., Sukop, M., 2007. Using of multiagent systems in robosoccer. Acta Mechanica Slovaca 11/2A, p. 157-160.
[5] Hajduk, M., Sukop, M., 2009. Multiagents system with dynamic box change for MiroSot. Progress in Robotics : Communications in Computer and Information Science 44, p. 287-292.
[6] Plaku, E., Kavraki, E. E., Vardi, M. Y., 2010. Motion planning with dynamics by a synergistic combination of layers of planning. Robotics, IEEE Transactions on, 26(3), p. 469-482.
[7] Al-Sabban, W. H., Gonzalez, L. F., Smith, R. N., Wyeth, G. F., 2012. Wind-energy based path planning for electric unmanned aerial vehicles using markov decision processes. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE.
[8] Konar, A., Goswami Chakraborty, I., Singh, S. J., Jain, L. C., Nagar, A. K., 2013. A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot.
[9] Zhu, M., Otte, M., Chaudhari, P., Frazzoli, E., 2014. Game theoretic controller synthesis for multi-robot motion planning Part I: Trajectory based algorithms. arXiv preprint arXiv:1402.2708.
[10] Chen, C. Y., Shih, B. Y., Shih, C. H.,Wang, L. H., 2013. Enhancing robust and stability control of a humanoid biped robot: system identification approach. Journal of Vibration and Control, 19(8), pp. 1199-1207.
[11] Barkaoui, M., Berger, J., Boukhtouta, A., 2014. An information-theoretic-based evolutionary approach for the dynamic search path planning problem. Advanced Logistics and Transport (ICALT), 2014 IEEE International Conference on, pp. 7-12..
[12] Ulusoy, A., Smith, S. L., Ding, X. C., Belta, C., 2012. Robust multi-robot optimal path planning with temporal logic constraints. Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 4693-4698).
[13] Gmiterko, A., Kelemen, M., Virgala, I., Surovec, R., Vacková, M., 2011. Modeling of a snake-like robot rectilinear motion and requirements for its actuators. INES 2011 : 15th IEEE International Conference on Intelligent Engineering Systems, Poprad, Slovakia, pp. 91-94.
[14] Bry, A., Roy, N, 2011. Rapidly-exploring random belief trees for motion planning under uncertainty. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 723-730.
[15] Plaku, E., Hager, G. D., 2010. Sampling-based motion and symbolic action planning with geometric and differential constraints. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 5002-5008.
[16] Surovec, R., Kelemen, M., Vacková, M., Virgala, I., 2011. A conceptual design of the self-reconfigurable mobile robot. ATP Journal plus, vol. 1, pp. 57-60.
[17] Virgala, I., Gmiterko, A., Kelemen, M., Surovec, R., Prada, E., Vacková, M., 2012. Snake-like robot motion analysis. ATP Journal plus, vol. 1, pp. 88-91.
[18] Wiemann, T., Lingemann, K., Hertzberg, J., 2013. Automatic Map Creation For Environment Modelling In Robotic Simulators. ECMS, pp. 712-718.
[19] Luo, R. C., Lai, C. C., 2012. Enriched indoor map construction based on multisensor fusion approach for intelligent service robot. Industrial Electronics, IEEE Transactions on, 59(8), pp. 3135-3145.
[20] Raja, P., & Pugazhenthi, S., 2012. Optimal path planning of mobile robots: A review. International Journal of Physical Sciences, 7(9), pp. 1314-1320.
[21] Valencia, R., Andrade-Cetto, J., Porta, J. M., 2011. Path planning in belief space with Pose SLAM. In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 78-83.
[22] Sadat, S. A., Chutskoff, K., Jungic, D., Wawerla, J., Vaughan, R., 2014. Feature-Rich Path Planning for Robust Navigation of MAVs with Mono-SLAM. In Int. Conf. on Robotics and Automation (ICRA).
[23] Židek, K., Svetlík, J., Rigasová, E., 2012. Environment mapping and localization od mobile robotics. International Scientific Herald, pp. 272-280.
[24] Fossel, J., Hennes, D., Claes, D., Alers, S., Tuyls, K., 2013. OctoSLAM: A 3D mapping approach to situational awareness of unmanned aerial vehicles. In Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, pp. 179-188.
[25] Hertzberg, C., Wagner, R., Birbach, O., Hammer, T., Frese, U., 2011. Experiences in building a visual SLAM system from open source components. In ICRA, pp. 2644-2651.
[26] Nakhaeinia, D., Tang, S. H., Noor, S. M., Motlagh, O, 2011. A review of control architectures for autonomous navigation of mobile robots. International Journal of Physical Sciences, 6(2), pp. 169-174.
[27] Cui, S. G., Wang, H., Yang, L., 2012. A Simulation Study of A-star Algorithm for Robot Path Planning. 16th international conference on mechatronics technology, pp. 506-510.
[28] Duchoň, F., Huňady, D., Dekan, M., Babinec, A., 2013. Optimal navigation for mobile robot in known environment, Applied Mechanics and Materials, 282, pp. 33-38.
[29] Yap, P. K. Y., Burch, N., Holte, R. C., Schaeffer, J. , 2011. Any-Angle Path Planning for Computer Games. AIIDE.
[30] Nash, A., Koenig, S., 2013. Any-Angle Path Planning. AI Magazine, 34(4), 9.
[31] D. Harabor, 2012. Fast Pathfinding via Symmetry Breaking, available at: http://aigamedev.com/open/tutorial/symmetry-in-pathfinding/
[32] Harabor, D. D., Grastien, A., 2011. Online Graph Pruning for Pathfinding On Grid Maps. In AAAI.