

Expecting the Unexpected: Measure the Uncertainties for Mobile Robot Path Planning in Dynamic Environment

Yan Li and Brian Mac Namee and John Kelleher

Applied Intelligence Research Center, School of Computing,
Dublin Institute of Technology, Dublin, Ireland
<http://www.comp.dit.ie/aigroup/>

Abstract. Unexpected obstacles pose significant challenges to mobile robot navigation. In this paper we investigate how, based on the assumption that *unexpected* obstacles really follow patterns that can be exploited, a mobile robot can learn the locations within an environment that are likely to contain obstacles, and so plan optimal paths by avoiding these locations in subsequent navigation tasks. We propose the DUNC (Dynamically Updating Navigational Confidence) method to do this. We evaluate the performance of the DUNC method by comparing it with existing methods in a large number of randomly generated simulated test environments. Our evaluations show that, by learning the likely locations of *unexpected* obstacles, the DUNC method can plan more efficient paths than existing approaches to this problem.

Keywords: Mobile robot navigation, dynamic environments, learning

1 Introduction

A basic requirement for a mobile robot is the ability to plan and execute a path to a goal destination [1, 2]. Unexpected obstacles, however, pose significant challenges to autonomous mobile robot navigation. For the purposes of this work we define unexpected obstacles as obstacles that may block corridors during robot navigation, but are not permanent features of the environment and hence do not appear on maps of the environment. For example, in a school building a robot may be trying to navigate a corridor and find that the corridor is blocked by a group of students who have just spilled out of a classroom. In situations where an unexpected obstacle blocks a path a robot may be forced to replan the route to the goal and take a detour, resulting in extra navigational costs in terms of time and energy.

Often, however, there is a regularity to the appearance of *unexpected* obstacles. For example, students may block a path at regular intervals (before and after their scheduled classes) or particular doors may be closed at particular times for security reasons. If a robot could learn the pattern of unexpected obstacles and annotate the map used for planning with this information then more efficient paths could be planned.

Contributions: The main goal of this paper is to investigate how a mobile robot can learn features of unexpected obstacles in dynamic environments, so that optimal shortest paths can be planned. In particular, we will develop and evaluate a method for annotating topological maps with information regarding the probability of a corridor being blocked based on the robot’s prior experiences in the environment.

Overview: This paper is organised as follows: In Section 2, we review background work. In Section 3, we describe our method for dynamically updating and maintaining the navigational confidence associated with an edge in a topological map, which we name DUNC. In Section 4, we present experiments and results on evaluating DUNC method. Finally, in Section 5, we discuss the performance of the DUNC method and suggest directions for future work.

2 Background

Path planning is a process of planning a collision-free path between a start position and a goal position [3]. It is a critical problem in mobile robot navigation. There are generally two types of robot path planning: static path planning and dynamic path planning [4, 5]. Static path planning finds optimal passable paths based on a full knowledge of the environment [6–8]. However, static path planning cannot be used to deal with unexpected obstacles in dynamic environments.

Dynamic path planning generates passable paths based on environment changes (i.e, unexpected closed doors). There are existing methods that deal with unexpected obstacles locally. Examples include potential field methods [11], the dynamic window approach [12] and vector field histograms [13]. However, these local planning techniques focus on real time local efficiency instead of global optimal path planning [14] and suffer from the problem of local minima [15]. As a result they have no mechanism for retaining the information gleaned from one navigation (for example the fact that the robot encountered an unexpected obstacle at a particular location) and using it to improve subsequent navigations. Some other existing approaches deal with uncertainties based on probabilistic methods for improving robot localisation, but they are not proved to be applicable to dynamic path planning problems [9, 10].

One existing dynamic global approach is proposed by Yamauchi and Beer [2]. They use the term *topological change* to describe variations in the environment that effect the decisions of a robot path planner. In their topological representation, a confidence value, defined as a “*robot’s certainty that a topological link can be traversed*”, is associated with every link. In their approach, if a robot successfully navigates from a topological node A to a topological node B , the navigational confidence associated with the link between A and B , C_{AB} , in the map is increased. If, however, the robot fails to reach B from A then C_{AB} is decreased. During path planning the cost of each candidate path is the sum of the costs associated with each link in the plan and the cost of each link C is a function of the navigational confidence associated with that link. One of the strengths of Yamauchi and Beer’s method is that successful navigation tasks are

recorded. But their method suffers from the fact that path cost is solely based on confidence and neglects other criteria such as path length. As such, their approach would preference a very long path over a much shorter path if the longer path had a slightly higher confidence associated with it. In our approach, we take the length of path into account to measure the uncertainty cost of each link.

Stentz and Hebert [16] propose an autonomous navigation system that deals with the cost of potential obstacle positions based on a predefined coefficient value (a scalar value σ). In their system, potential obstacle positions are marked with high scalar values, so that the path planner will avoid planning paths across those positions. Then a graph search method is used to find an optimal path with the least cost. The scalar method suffers from the problem that since the cost of all the edges are updated based on the same σ , the cost update process has little effect on potential obstacle positions with low static cost and therefore a marked potential block position will be considered as unmarked by the path planner. In our approach, unexpected obstacles are measured separately for different edges and the measurements are normalised with maximum edge length.

Briggs et al. [17] propose a ratio method that deals with unexpected obstacles in dynamic environments based on a topological representation. In their approach, each topological edge is associated with a ratio value R . R is calculated as the ratio of the number of detections of unexpected obstacles and total number of traverses on the topological edge. After every edge traversal, the ratio value is updated based on whether an unexpected obstacle is detected or not. If the robot successfully traverses an edge, the ratio of the edge is decreased, otherwise R is increased. Hu and Brady [1] propose a similar method to Briggs et al's. They classify uncertainties as three types on a predefined path: obstacles that partially block the robot path, obstacles that fully block the robot path and moving obstacles that cross the robot path. They measure three types of uncertainties based on a ratio value R . Interestingly a decay process is used in Hu and Brady's approach to release edges with high R based on an exponential function for the subsequent navigation tasks. Essentially this decay process allows the system to forget that the robot encountered an obstacle on a particular edge and makes the edge viable during subsequent path planning. In contrast to Hu and Brady's approach, our approach uses a dynamic confidence scoring method with a confidence decay component to measure the uncertainties in dynamic environments.

3 The DUNC Method

In this section we present our method for dynamically updating the navigational confidence associated with edges in a topological map, which we will refer to as the DUNC method. The DUNC method is part of our Dynamic Confidence Topological Map Approach (*DCTMA*). *DCTMA* is an approach for solving dynamic robot path planning problems based on a topological representation of the

world. It is built on top of our landmark-based robot localisation and navigation system [18], with corridor following and obstacle avoidance behaviours.

A Dynamic Confidence Topological Map $DCTM = \langle N, E \rangle$ is a topological map that can be maintained and updated dynamically based on the sensor data of the robot as it moves through the environment. A $DCTM$ contains a set of nodes $N = \{N_1, N_2, \dots, N_n\}$, which represent landmarks (i.e., T-junctions, X-junctions) in the environment. And it contains a set of edges $E = \{E_1, E_2, \dots, E_n\}$, which are passable edges between landmark positions. Each edge has a 3-tuple data structure $\{D, L, C\}$, where: D is the static cost of the edge (i.e., the length of an edge); L is running score that is increased by a fixed amount, δ_+ , each time the edge is successfully navigated (see Equation 1) and decreased by a fixed amount, δ_- , each time the robot encounters an obstacle on the edge (see Equation 2); and C is the confidence that an edge can be traversed by the robot and is always calculated from L using Equation 3. In the DCTM system L is initialised to 0 and C is consequently initialised to 0.5.

$$L = L + \delta_+ \quad (1)$$

$$L = L - \delta_- \quad (2)$$

$$C = 1 - \frac{1}{1 + e^L} \quad (3)$$

During path planning the cost of each edge E_x is calculated based on D and C via a weighted sum aggregation function given W_D and W_C as the corresponding weight for D and C . In this system the values of W_D and W_C are constrained such that the sum of the weights, $W_D + W_C$, always equals 1. Then E_x is used to feed the Dijkstra graph search method [19] to find optimal paths.

As stated in Equation 3, the confidence score of an edge is a function of the L score of the edge which has the value range from $-\infty$ to ∞ . This has the potential to cause problems as the L score for a particular edge may become so low that paths including the edge are never considered for navigation or so high that paths including the edge are always selected even if the robot encountered an obstacle on that edge very recently.

One way of addressing this issue is to impose a bounds on the range of values that L can take. We have found that bound the range of L between -4 and $+4$ permits a confidence range (0.018 to 0.982) that is large enough to reflect the navigation results (encounter an obstacle or successful traverse) while at the same improving the chance that a low confidence edge will be reused at limit points.

Another extension that could be made to the basic confidence representation is inspired by Hu and Brady's decay process [1]. In the DUNC method the proposed memory decay process implements a regression towards the initial a confidence value of 0.5 on the edges in the $DCTM$. The memory decay component makes the robot periodically regress the L score of an edge (and hence the edge confidence) towards the initial value of 0; essentially, allowing the robot to forget the obstacles it has encountered on the edge or its previous successful

navigations of the edge. The motivation for proposing this extension is similar to the motivation for imposing a bounds on the range of L namely that as the robot navigates through the environment some edges in the *DCTM* may be marked with a low confidence, due to the robot failing to navigate through them successfully. The effect of marking an edge with a low confidence is that that edge has a low probability of being used for subsequent navigation tasks. The memory decay component releases these edges to make them accessible again to the robot. Also, it decreases L on high confidence edges to avoid persistent usage of high confidence edges. The memory decay component updates the log odds L on each edge based on a parameter ν , which is the frequency of activating the memory decay component. In our system, memory decay component is switched on every ν robot navigation tasks. When the memory decay process is run it increase the L value of every edge with a confidence value lower than 0.5 by a constant value, ρ_+ , in the range 0:1, (see Equation 4) and decreases the L value of every edge with a confidence value greater than 0.5 by a constant value, ρ_- , in the range 0:1, (see Equation 5).

$$L = L + \rho_+ \quad (4)$$

$$L = L - \rho_- \quad (5)$$

In summary, the DUNC method dynamically updates the navigation confidence associated with an edge in a topological map representation based on the robots experiences in navigating along the edge. The basic component of the method is a logistic confidence model. There are two proposed extensions to this model and bounding of the L values and a memory decay process. In the next section we present a series of experiments that evaluate the contribution of each of these components to the method and compare the method to the current state-of-the-art methods.

4 Evaluation

This section describes experiments performed in order to evaluate the performance of the DUNC method. First the test environment used and the procedure used to generate *unexpected* obstacles within this environment will be described. Then we will describe 3 different experiments that evaluate the DUNC method and compare it to existing methods discussed in Section 2.

4.1 Generating Test Environments

All of the experiments that we describe take place in simulated environments with simulated robots (any type of robot can be used). In order to run experiments evaluating the performance of the DUNC method we generate random indoor environments (represented as topological maps) in which robot navigations are simulated. The simulated robot actually jumps between topological

nodes and the navigation distance is measured as the actual distance between the start node and the end node along topological edges. Each random environment is composed of 20 to 40 nodes connected by 40 to 80 edges. The nodes to be connected to each other are randomly selected. Figure 1 shows a representation of one such environment.

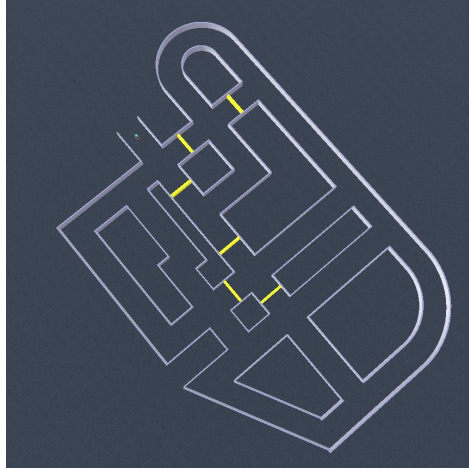


Fig. 1. An example of a randomly generated simulated test environment. Potential obstacle positions are marked with yellow bars.

To determine which edges in a map will periodically become blocked by obstacles, we first simulate 1,000 journeys from random start nodes to random end nodes within this map without any unexpected obstacles being present. For each journey, the Dijkstra algorithm is applied given its start node and end node. The result of the Dijkstra algorithm is a path consist of a set of topological nodes. Navigations are simulated based on the results of Dijkstra algorithm to count the number of times each edge in the map is used in these journeys. Then, the edges that are used most frequently (the top 25% in our experiments) are selected as potential obstacle positions. In this way we introduced obstacles into the most important edges within the environment. In Figure 1 the potential blockage positions are marked as yellow rectangles. Each of the potential blockage positions has a randomly selected *obstacle appearance rate* associated with it. This is the probability (from 0 to 1) that an obstacle will appear on an edge when a robot tries to navigate it. Obstacle appearance rates remained constant throughout each experiment in the corresponding environmental unit.

4.2 Experimental Method

Each experimental unit consisted of a robot performing 1,000 journeys between randomly selected start and end nodes in a test environment. Based on the

obstacle appearance rates associated with each edge in the map different edges would become blocked by obstacles during each of these journeys. When multiple techniques are compared, the same 1,000 journeys are always performed, during which the same edges are blocked. In this way different techniques are fairly compared.

When an unexpected obstacle is encountered during a navigation task, the robot will be forced to re-plan the route to the goal and detour to avoid the obstacle. As a result, each unexpected obstacle encountered will result in extra travel cost (extra travel distance in our experiments). For each journey a robot takes from a start node to an end node we record the length of the *actual path* it navigates, which is the summation of lengths of edges the robot traversed. We also record the length the *optimal path* would have been if the robot had known in advance where all the unexpected obstacles would occur. The key metric used in our experiments is the difference between the actual path length and the optimal path length. The average difference between these two measures is calculated across 1000 journeys for each experimental unit.

In the remainder of this section, we present 3 experiments that (1) fine tune the DUNC method, (2) find the best parameter values to use for the existing methods described in Section 2, and (3) compare the performance of the DUNC method with these existing methods.

4.3 Experiment 1: Fine Tuning the DUNC Method

The DUNC method is composed of three components: (1) the confidence representation itself, (2) the bounds component, and (3) the memory decay component. Thus, there are four candidate combinations of these components:

- confidence representation (*confidence*)
- confidence representation + bounds component (*confidence* + B)
- confidence representation + memory decay component (*confidence* + D)
- confidence representation + bounds component + memory decay component (*confidence* + B + D)

Each of the component parts of the DUNC method uses an associated set of parameters. Table 1 lists these parameters and, for each, the set of potential values they can assume. Table 2 shows parameter values that are used by each of the candidate combinations of the components of the DUNC method.

The first experiment sought to determine which was the most effective out of the four combinations of DUNC components, i.e. *confidence*, *confidence* + B , *confidence* + D and *confidence* + B + D . In these experiments the *best* parameter values shown in Table 2 were used. Robots using each of the four combinations of DUNC components were simulated performing 1,000 navigations each across 50 randomly generated environments (in all each robot performed 50,000 navigations). For each of the 50 environments the average difference between actual and optimal path lengths was recorded for each combination of DUNC components. The DUNC component combinations were then ranked based on their average

Parameter	Description	Values
ω	the weight of the distance component used by the weighted sum cost function	10, 20, 30, 40, 50, 60, 70, 80, 90
ν	the frequency at which the memory decay component is invoked	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
$\delta+$	the amount by which the running score is increased after a successful navigation	0.5, 1
$\delta-$	the amount by which the running score value is decreased after an unsuccessful navigation	0.5, 1
$\rho+$	the amount by which to update the running score value by the memory decay component on low log odds edges	0,0.5,1.0
$\rho-$	the amount by which to update the running score value by the memory decay component on high log odds edges	0.5, 1

Table 1. The parameters used by the DUNC method and the possible values they can assume

Combinations	ω	ν	$\delta+$	$\delta-$	$\rho+$	$\rho-$
<i>confidence</i>	50		0.5	1		
<i>confidence</i> + B	40		0.5	1		
<i>confidence</i> + D	50	50	0.5	1	0	0.5
<i>confidence</i> + B + D	40	100	0.5	1	0.5	0.5

Table 2. Best parameter values for DUNC component combinations

difference between actual and optimal path lengths on each environment (ranks ranged from 1 (*best*) to 4 (*worst*)). Table 3 shows the average rank achieved across the 50 test environments by each DUNC component combination. From this it is clear that the *confidence* + B + D combination appears to perform the best, closely followed by *confidence* + B combination.

To test these results for statistical significance, we applied the Friedman test [20] followed by the Holm step-down procedure [21] as a post-hoc test on the average ranks. The results of the Friedman test show that there is a statistically significant difference in robot performance depending on which combination of DUNC components was used, $X^2_F(3) = 24.1446$, $p = < 0.0001$.

	<i>confidence</i>	<i>confidence</i> + B	<i>confidence</i> + D	<i>confidence</i> + B + D
Avg Rank	2.84	2.12	3.04	2

Table 3. Average ranks of candidate DUNC component combinations calculated for the Friedman test.

In order to investigate significant differences between the best DUNC component combination, $confidence + B + D$, and the others we applied the Holm’s step-down procedure as a post-hoc test to the Friedman test. Table 4 shows the result of this test. By comparing the p values in the second column to the critical values in the third column¹ we can see a statistically significant difference exists between the $confidence + B + D$ and the $confidence$ and $confidence + D$ combinations but not the $confidence + B$ combination. This is not surprising given that the average ranks of the $confidence + B + D$ and $confidence + B$ combinations were so close.

<i>Approaches</i>	z	p	$\alpha/(k - i)$
<i>confidence + D</i>	4.0279	0.00056	0.017
<i>confidence</i>	3.2533	0.001	0.025
<i>confidence + B</i>	0.4648	0.642	0.05

Table 4. Results of the Holm step-down procedure.

Based on these experiment results, we see that the memory decay component does not really help the performance of the DUNC method. We believe that the reason for this is that, as long as the likelihood of an edge being blocked by an obstacle remains constant (as is the case in our experiments), *forgetting* about the existence of obstacles lends advantage. This is also evident in the very poor performance of the $confidence + D$ combination. The bounds component, however, has a significant impact on performance. We surmise that this is because it allows the update procedure remain agile and respond to successful and unsuccessful edge navigations quickly. In the remaining experiments when we refer to the DUNC method we refer to the $confidence + B$ component combination as, adopting the principle of *parsimony*, it makes sense to choose a slightly less complex method given almost equal performance.

4.4 Experiment 2: Fine Tuning Existing Methods

So as the comparison between existing methods and the DUNC method will be fair, we perform the same fine tuning experiment for the existing methods described in Section 2: the ratio method (*Ratio*), the ratio method with decay process (*RatioDecay*), the scalar method (*Scalar*) and the Yamauchi method (*Yamauchi*). Each of these methods (excluding the ratio method) has a set of parameters for which optimal values must be discovered. Table 5 shows the candidate methods and their corresponding parameters and possible values that are considered. Table 6 shows the resulting *best* parameter values and these are used in the final experiment described in this section.

¹ In a Holms step-down procedure a statistically significant difference exists when the p value is less than the critical value.

Method Name	Parameters	Possible Values
<i>Ratio</i>	No parameter used	
<i>RatioDecay</i>	Decay rate μ	0.005,0.025,0.05,0.1,0.5,1,2,5
	Decay Frequency Δ	1,2,3,4,5,10,20,50,100
<i>Scalar</i>	Scalar value σ	2,3,4,5,10,15,20,50,100
<i>Yamauchi</i>	Link learning rate λ	0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9

Table 5. Existing methods and their corresponding parameter and possible values.

Method Name	Parameter Values
<i>RatioDecay</i>	Decay rate $\mu = 0.005$
	Decay Frequency $\Delta = 100$
<i>Scalar</i>	Scalar value $\sigma = 5$
<i>Yamauchi</i>	Link learning rate $\lambda = 0.1$

Table 6. Best parameter values for the *RatioDecay*, *Scalar* and *Yamauchi* methods.

4.5 Experiment 3: Comparing the DUNC Method to Existing Approaches

The aim of this experiment is to compare the DUNC method to existing approaches that deal with unexpected obstacles in robot navigation across dynamic indoor environments. In this experiment we compare the performance of the DUNC method, the ratio method, the ratio method with decay process, the scalar method and the Yamauchi method. In this experiment robots using each of these methods were simulated performing the same 1,000 journeys across 50 randomly generated environments (50,000 journeys in total per robot). The average differences between actual and optimal paths for each of these sets of journeys were recorded. the performance of the 5 different methods were ranked (from 1 (*best*) to 5 (*worst*)) across the different random environments and the average ranks are shown in Table 7. It is clear from these ranks that the DUNC approach appears to choose more efficient routes than any of the other approaches.

	<i>DUNC</i>	<i>Scalar</i>	<i>Ratio</i>	<i>RatioDecay</i>	<i>Yamauchi</i>
Avg Rank	1.18	2.4	3.66	4.32	3.44

Table 7. Average rank values of candidate methods calculated for the Friedman test.

In order to test for statistically significant differences in these results a Friedman test was used again. The results of the Friedman test show that there is a statistically significant difference in robot performance depending on which approach is used, $X_F^2(4) = 120.88$, $p < 0.0001$.

In order to further investigate this result we applied the Holms step-down procedure again. Table 8 shows the result of this analysis comparing the DUNC method with the other approaches. Again, by comparing the p -values in the

second column with the critical values in the third column we can see that a statistically significant difference exists between the performance of the DUNC method and all of the other approaches. This result shows that, under our test conditions, a robot using the DUNC method to navigate within an indoor environment selects more efficient paths than a robot navigating using any of the other methods under test.

Methods	z	p	$\alpha/(k-i)$
<i>RatioDecay</i>	9.93	< 0.0001	0.125
<i>Ratio</i>	7.842	< 0.0001	0.0167
<i>Scalar</i>	7.148	< 0.0001	0.025
<i>Yamauchi</i>	3.858	0.00011	0.05

Table 8. Results of the Holm step-down procedure.

5 Conclusion

In this paper we presented the DUNC method that a mobile robot can use to learn likely positions of unexpected obstacles in dynamic environments, so that optimal shortest paths can be planned. This is built on the assumption that most *unexpected* obstacles are not really unexpected at all and follow a pattern occurring in similar places over time and that such patterns can be exploited. The DUNC method is composed of three key components: the confidence component and a set of bounds on running outcome scores (a forgetting mechanism was considered for inclusions but tests showed that it did not improve overall performance).

The DUNC method was compared against the most important existing methods for addressing this problem - the ratio method, the ratio method with decay process, the scalar method and the Yamauchi method. The DUNC method was found to consistently plan more efficient paths than all of these other approaches across a large number of test runs in different environments.

There are a range of ways in which we could improve the DUNC method. In particular, in the future we plan to examine how the approach can be modified to handle partial blockages of routes (currently all obstacles completely block an edge on the map), different ways the decay mechanism could be adjusted to make it more effective, ways in which the approach could be modified to handle moving obstacles and the introduction of a time dimension in which different routes are assumed to be more likely to be blocked at different times of the day.

References

1. Hu, H., Brady, M.: Dynamic Global Path Planning with uncertainty for mobile robots in manufacturing. IEEE International Conference on Robotics and Automation, pp. 760–767. (1997)

2. Yamauchi, B., Beer, R.: Spatial Learning for Navigation in Dynamic Environments. In: Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, pp. 496–505. (1996)
3. Hsu, D., Latombe, J.C., Motwani, R.: Path planning in Expansive Configuration Spaces. IEEE International Conference on Robotics and Automation. (1997)
4. Phillips, B., Likhachev, M.: SIPP: Safe Interval Path Planning for Dynamic Environments. IEEE International Conference on Robotics and Automation, pp. 9–13. Shanghai (2011)
5. Wang, T., Dang, Q., Pan, P.: Path Planning Approach in Unknown Environment. International Journal of Automation and Computing, 7, 310. (2010)
6. Hao, G., Zhang, D., Feng, X.: Model and Algorithm for Shortest Path of Multiple Objectives. Journal of Southwest Jiaotong University. 42, 641–646. (2007)
7. Kulyukin, V., Kutianawala, A., LoPresti, E., Matthews, J., Simpson, R.: iWalker: Toward a Rollator-Mounted Wayfinding System for the Elderly. IEEE International Conference on RFID. (2008)
8. Murphy, L., Newman, P.: Planning Most-Likely Paths From Overhead Imagery. IEEE International Conference on Robotics and Automation (ICRA). (2010)
9. Mathibela, B., Osborne, M.A., Posner, I., Newman, P.: Can Priors Be Trusted? Learning to Anticipate Roadworks. 15th International IEEE Conference on Intelligent Transportation Systems (ITSC). (2012)
10. Wolf, D.F., Sukhatme, G.S.: Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments. Journal of Autonomous Robots, 19, 53–65. (2005)
11. Koren, Y., Borenstein, J.: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. IEEE International Conference on Robotics and Automation. (1991)
12. Baehoon, C., Beomseong, K., Euntai, K., Kwang-Woong, Y.: A Modified Dynamic Window Approach in Crowded Indoor Environment for Intelligent Transport Robot. International Conference on Control, Automation and Systems. (2012)
13. Borenstein, J., Koren, Y.: The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. IEEE International Conference on Robotics and Automation. 7, 278–288. (1991)
14. Ding, F., Jiao, P., Bai, X., Wang, H.: AUV Local Path Planning based on Virtual Potential Field. IEEE International Conference on Mechatronics and Automation. (2005)
15. Majdi, M., Anvar, H.S., Barzamini, R., Soleimanpour, S.: Multi AGV Path Planning in Unknown Environment using Fuzzy Inference Systems. International Symposium on Communications, Control and Signal Processing. (2008)
16. Stentz, A., Hebert, M.: A Complete Navigation System for Goal Acquisition in Unknown Environments. Autonomous Robots, 2, 127–145. (1995)
17. Briggs, A. J., Detweiler, C., Scharstein, D., Vandenberg-rodes, A.: Expected Shortest Paths for Landmark-Based Robot Navigation. International Journal of Robotics Research, 23, 717–728. (2004)
18. Li, Y., Mac Namee, B., Kelleher, J.D.: Navigating the Corridors of Power: Using RFID and Compass Sensors for Robot Localisation and Navigation. In: the 11th Towards Autonomous Robotic Systems (TAROS 2010). (2010)
19. Edsger W. D.: A Note on Two Problems in Connexion with Graphs. Numerische Mathematik. 1, 269–271. (1959)
20. Friedman, M.: A Comparison of Alternative Tests of Significance for the Problem of m Rankings. The Annals of Mathematical Statistics. 11, 86–92. (1940)
21. Holm, S.: A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics. 6, 65–70. (1979)