International Journal of Control, Automation and Systems 16(6) (2018) 3026-3035
http://dx.doi.org/10.1007/s12555-018-0037-6
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Fuzzy Greedy RRT Path Planning Algorithm in a Complex Configuration Space

Ehsan Taheri*, Mohammad Hossein Ferdowsi, and Mohammad Danesh

Abstract: A randomized sampling-based path planning algorithm for holonomic mobile robots in complex configuration spaces is proposed in this article. A complex configuration space for path planning algorithms may cause different environmental constraints including the convex/concave obstacles, narrow passages, maze-like spaces and cluttered obstacles. The number of vertices and edges of a search tree for path planning in these configuration spaces would increase through the conventional randomized sampling-based algorithm leading to exacerbation of computational complexity and required runtime. The proposed path planning algorithm is named fuzzy greedy rapidly-exploring random tree (FG-RRT). The FG-RRT is equipped with a fuzzy inference system (FIS) consisting of two inputs, one output and nine rules. The first input is a Euclidean function applied in evaluating the quantity of selected parent vertex. The second input is a metaheuristic function applied in evaluating the quality of selected parent vertex. The output indicates the competency of the selected parent vertex for generating a random offspring vertex. This algorithm controls the tree edges growth direction and density in different places of the configuration space concurrently. The proposed method is implemented on a Single Board Computer (SBC) through the xPC Target to evaluate this algorithm. For this purpose four test-cases are designed with different complexity. The results of the Processor-in-the-Loop (PIL) tests indicate that FG-RRT algorithm reduces the required runtime and computational complexity in comparison with the conventional and greedy RRT through fewer number of vertices in planning an initial path in significant manner.

Keywords: Holonomic robot, processor-in-the-loop test, rapidly-exploring random tree, sampling-based path planning, single board computer.

1. INTRODUCTION

The efficiency of autonomous unmanned vehicles (AUxV) for scientific, commercial and military operations have increased over the past two decades. This in turn, has made more researchers to run more studies thereof. One of the biggest available challenging bottlenecks in developing these robots is the technology readiness level in the autonomy field [1]. In order to increase the level of autonomy in these robots, at least, cooperation of four subsystems including guidance, control, navigation and path planning is required [2]. Running studies on path planning algorithm is one of the open research areas for holonomic and non-holonomic autonomous robots [3]. The path planning ability is defined as a skill in AUxV to convert the high-level human demand (mission) to the low-level human demand (displacement) [4]. In this context, the planning collision-free paths among polyhedral obstacles is accomplished for the first time by [5] which is considered as a milestone in the development of this field. For assessing this subject, authors in [6] have reviewed about 200 articles and authors in [7] have reviewed about 120 articles. Path planning issues are developed through the two main groups of classical and heuristic candidate algorithms: Classical algorithms consist of: Bugs [8], Voronoi Diagram (VD) [9], Visibility Graph (VG) [10], Cell Decomposition (CD) [11] and Potential Fields [12]. In this group, the geometric or kinematic aspect of paths is the main issue where the kinodynamic constraints and the caused movement factor in the robot are not of concern. These algorithms are efficient in path planning, provided that, all degrees of freedom of a robot are independently controllable or the robot has a small inertia. Classical algorithms are deterministic and complete where their performances are usually evaluated through the piano mover's problem benchmark [13]. The main drawbacks in this group consist of: 1) sensitivity to parameter tuning, 2) exponential increase in computational complexity and required runtime due to an increase in dimensionality of the configuration space and the density of obstacles, 3)

Manuscript received January 19, 2018; revised May 19, 2018; accepted June 26, 2018. Recommended by Associate Editor DaeEun Kim under the direction of Editor Euntai Kim.

Ehsan Taheri and Mohamad Hossien Ferdowsi are with the Control Group, Electrical Engineering Department, Malek-Ashtar University of Technology, Tehran 15875-1774, Iran (e-mails: taheri.ehsan@mut-es.ac.ir, Ferdowsi@mut.ac.ir). Mohammad Danesh is with the Department of Mechanical Engineering, Isfahan University of Technology, 84156-83111, Isfahan, Iran (e-mail: danesh@cc.iut.ac.ir).
* Corresponding author.

requiring a precise map of workspace, 4) existence of local minimums and 5) inability in applying the variables such as linear and angular velocities in path planning procedure. In order to overcome these drawbacks, the workspace is transferred from the configuration space to state space, while in the state space, the computational complexity and required runtime increase greatly because the path planning variables double in number.

Heuristic algorithms: some of these are developed based on randomized sampling method. This group is probabilistically complete. In probabilistic complete algorithms, if a path exists and time is infinite, then the path design is certain. However, if there exists no path, the probabilistic complete algorithm will not become converged to its solution [14]. The major advantages of sampling-based algorithms in path planning consist of: 1) the internal (differential) and external (kinematic) constraints of the robot are applied in the sampling process, 2) the state variables are replaced by configuration variables in a sense that changing variables rate can be considered in the generation of random vertices, 3) the collision detection module is applied as a black box in sampling-based algorithm, 4) a free configuration space is not necessary to be constructed in a high dimensional space for robots with high degrees of freedom and 5) the initial path is designed rapidly through the sampling-based algorithms. The main sampling-based path planning algorithms include: Probabilistic Roadmap (PRM), Rapidly Exploring Dense Tree (RDT), Rapidly-exploring Random Tree (RRT) and Expansive-Spaces Tree (EST). The PRM algorithm is proposed for path planning in high-dimensional configuration spaces by [15]. This algorithm consists of the two training and query phases and is appropriate for multiple-query problems. The training phase is accomplished offline, while query phase is accomplished online. In this study the RRT path planning algorithm is of concern. RRT is presented by [16], which is of only one phase, accomplished online. This algorithm is perfect for single query issues [17]. RRT is based on the search tree ,while PRM is based on the graph (roadmap). RRT is popular method for rapid path planning in highdimensional spaces and its different procedure are being developed for holonomic and non-holonomic robots as: Informed RRT* [18], RRT*-smart [19], LQR-RRT* [20], RRT-connect [21, 22],Kinodynamic RRT* [23], anytime RRT [24], skilled-RRT [25], Theta*-RRT [26], BI2RRT* [27], Cloud RRT* [28], bidirectional RRT* [29], RRT*-AB [30], MDMI-RRT* [31], LBT-RRT [32],T-RRT [33], Line Segment [34], Reverse Time Tree [35], Anytime Synchronized-Biased-Greedy RRT [36], DT-RRT [37]. Among the most RRT algorithms mentioned above, the Euclidean metric function is commonly applied for measurement distance between the vertices and edge lengths. The ability of this metric function in complex configuration space which include the convex/concave obstacles, narrow passage, maze like spaces, cluttered obstacles is insufficient and search tree may be trapped into a local minimum. The tree edges (branches) density increase in an inefficient manner in inappropriate places of configuration space through this metric function. For this purpose the FG-RRT algorithm is proposed. The main contributions of this article are briefed as follows:

1) Density of the selected parent vertices in FG-RRT are controlled through the metaheuristic function. In this manner, density of the search tree edges (branches) and vertices increase in appropriate places in the configuration space, where the total numbers of vertices and edges are reduced to plan an initial path, hence computational complexity and required runtime are decreased. It is notable that the density of search tree edges and vertices in the conventional RRT and greedy RRT increase in different places of the configuration space randomly.
2) Growth directions of the search tree in the FG-RRT are guided by the Euclidean function. In this manner, search tree grows up in appropriate directions. It is notable that in the conventional RRT the search tree grows up in random directions.
3) The FG-RRT is able to make decision through FIS. If selected parent vertex is of sufficient competency, a random offspring vertex is generated to expand and explore in appropriate places in the configuration space, otherwise, a new parent vertex is selected. Moreover, FG-RRT is implemented on the Axiomtek SBC 84710 through the xPC Target and then evaluated through PIL test. For this purpose, four test-cases are designed. The PIL tests results indicate that the required runtime and computational complexity in FG-RRT are reduced in comparison with the conventional and greedy RRT.

This article is organized as follows: a brief background of the RRT path planning algorithm is explained in Section 2; the proposed FG-RRT algorithm is presented and then implemented through the xPC Target on a SBC for evaluating with the PIL test in Section 3; performance and effectiveness of the proposed method is compared with the conventional and greedy RRT in path planning by four test-cases with different complexity in Section 4 and finally the article is concluded in Section 5.

2. BACKGROUND OF RRT PATH PLANNING ALGORITHM

This algorithm is heuristic, which due to its high execution speed, simplicity and limited amount of computations is appropriate for real-time applications in complex configuration spaces. The structure of RRT algorithm consists five main functions of: 1) metric, 2) random sampling, 3)

steer branch, 4) nearest neighbors and 5) collision detection functions [16].

2.1. Metric function

The proper selection of metric function affects the ability of sampling-based algorithms [38]. For this purpose, as shown in Fig. 1 if the Euclidean metric function is applied, the path planning algorithm will act in an appropriate manner, while due to the kinodynamic constraints of vehicle and restriction of side motion, designing a feasible path between the initial and target points would not be possible. The metric function (ρ) is defined as:

$$I_p = \rho(x,x') = (\sum_{i=1}^{n} |x_i - x'_i|^p)^{\frac{1}{p}}. \quad (1)$$

If (p) in (1) is 1, 2 or ∞ the metric function is named Manhattan, Euclidean or Chessboard, respectively.

2.2. Random sampling function

A random configuration variable (q_{Rand}) is generated through this function followed by adding a new vertex (q_{New}) to the search tree $T(V,E)$ for exploring in the configuration space. In this context there exists many sampling strategies applicable in a search tree as: 1) sampling around the M-Line axis in the BUG algorithm, 2) sampling around the Medial axis in the PRM algorithm, 3) sampling in large areas of VG and VD, 4) applying the boundary layer and Gaussian techniques to increase the probability of sampling at the border of the obstacles, 5) running the bridge test to increase the probability of sampling in narrow passages and 6) adding bias to the sampling procedure [39].

2.3. Steer branch function

In this function, (q_{Rand}) and (q_{Parent}) variables are involved in designing a path (γ) between them in free configuration space (X_{Free}) as an output. Kinematic, dynamic,environmental, operational and the maximum distance of robot mobility in one step are applied in this function as the constraints. The edge of the search tree grows up from the place of the parent vertex in the configuration space to its target point of the edge which is a random vertex (q_{Rand}) or new vertex (q_{New}) based on the considered constraint [23].

2.4. Nearest neighbors function

A set of the best candidate vertices $(V_{Near} \subset V)$ are considered between all available vertices of search tree (V) through this function and then one of them would be selected to become a parent vertex (q_{Parent}) for generating a random offspring vertex (q_{Rand}). A set of the best candidate vertices (V_{Near}) is defined by:

$$V_{Near}$$

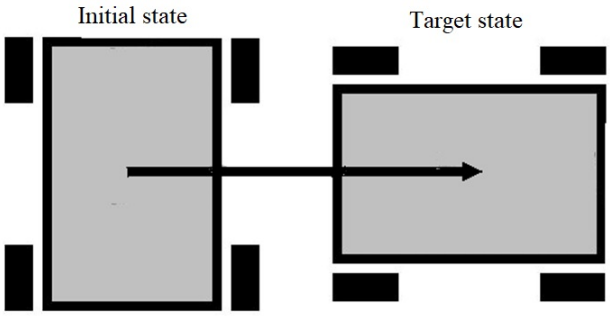

Fig. 1. Drawback of Euclidean metric function for path planning in the presence of kinodynamic constraints.

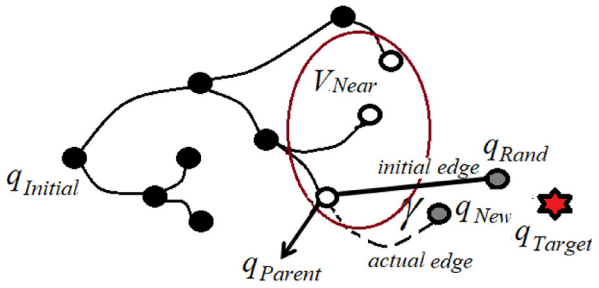

Fig. 2. Describing RRT path planning algorithm concept.

$$= \left\{ x_i \in V.(i = 0,...,K) : D(x,x_i) < \alpha(\frac{\log N}{N})^{\frac{1}{d}} \right\}, \quad (2)$$

where,d is the dimension of configuration space, N is the vertex count in the search tree,α is the design parameter and K is the neighboring vertex count.

2.5. Collision detection function

The output of the random sampling function(q_{Rand}) and branch function $(\gamma(x) \forall x \in x_{free}, q_{new})$ are evaluated by this function for located in free configuration space (X_{Free}).

Based on the above considered five functions the schema of this is shown in Fig. 2. The nearest neighbors to the random offspring vertex (q_{Rand}) are marked by empty circles and the subset of these neighbors is shown by a large circle(V_{Near}). The initial edge of search tree between the parent vertex (q_{Parent}) and random offspring vertex (q_{Rand}) is shown by a solid line. The actual edge of search tree is oriented from the parent vertex (q_{Parent}) to the new vertex (q_{New}) and is shown by dotted line. The discrepancy between the solid and dotted lines is due to either motion constraints or insufficient time.

3. PROPOSED FG-RRT

This FG-RRT is proposed to increase the computational runtime speed and to decrease computational complexity

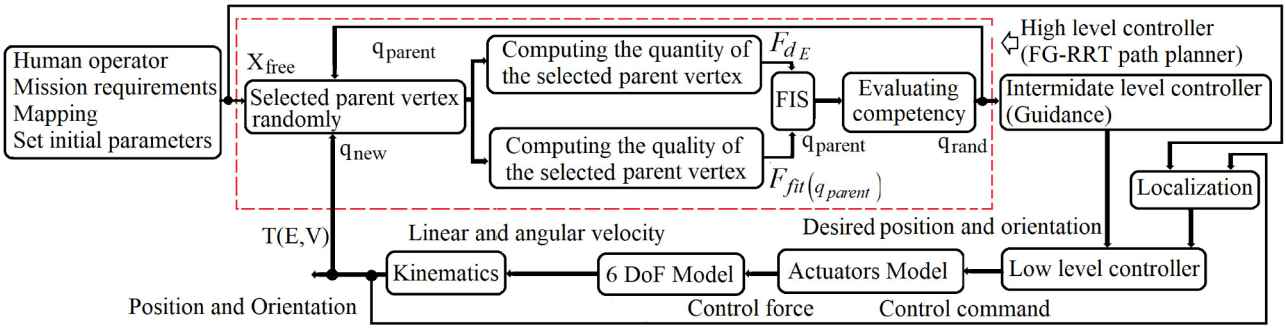

Fig. 3. The schematic block diagram of the FG-RRT path planning algorithm.

in complex configuration spaces. Here the quantity of the selected parent vertex is evaluated by the Euclidean function and its quality is evaluated by the metaheuristic function. The quantity term is the distance between the target point and selected parent vertex, presented in subsection 3.1, while quality term is the position of selected parent vertex in the free configuration space, presented in subsection 3.2. The outputs of these two functions (quantity and quality) are fed to the FIS and then competency of the selected parent vertex is evaluated for generating an offspring vertex. If the selected parent vertex has sufficient competency, a random offspring vertex is generated, otherwise, a new parent vertex is selected. For example in the FG-RRT when a selected parent vertex of the search tree is trapped near the target point in a local minimum, quantity is good but quality is bad and so competency of that is insufficient for generating a random offspring vertex. Through this method not only the tree edges growth direction and density are to be controllable but also fewer number of vertices and edges are required for path planning. Therefore, computational complexity is reduced and FG-RRT becomes applicable for real time application. Structure of FG-RRT path planning algorithm is shown in Fig. 3. First, initial parameters are determined, including: the free configuration space, initial and target vertices of search tree, nine fuzzy rules, maximum iteration for path planning and maximum iteration for parent vertex (q_{parent}) selection. Then, quality and quantity of the selected parent vertex are computed by metaheuristic function presented by (4) and by Euclidean function presented by (3), respectively. Density and growth direction of edges are controlled concurrently through the quality and quantity factor in the proposed FG-RRT. Outputs of these two functions are fuzzified through the membership functions, in Fig. 4(a)-(b). Fuzzy values enter FIS where competency of the selected parent vertex is computed. Finally, if the selected parent vertex is of sufficient competency, a random offspring vertex is generated and a new vertex (q_{New}) is added to the search tree $T(v,E)$ for exploring configuration space, otherwise, a new parent vertex is selected.

Table 1. Comparison the features of conventional RRT, greedy RRT and FG-RRT.

	Growth directions of the search tree	Density of the vertices and edges	Make decision ability
Conventional RRT	Randomly	Randomly	Unable
Greedy RRT	Controllable	Randomly	Unable
FG-RRT	Controllable	Controllable	Enable by using FIS

Features of the conventional RRT, greedy RRT and FG-RRT are tabulated and compared in Table 1.

3.1. First input

This input in FIS is obtained from the Euclidean function through (3):

$$E^n \equiv (R^n.d_E) \leftrightarrow R^n \times R^n, \tag{3}$$
$$d_E(x^{'}.x^{"}) = +\sqrt{(x_1^{'}-x_1^{"})^2+\ldots+(x_n^{'}-x_n^{"})^2},$$
$$\forall x^{'}x^{"} \in R^n,$$

where (E^n) is the metric space on a set (R) with Euclidean function (d_E). In order to fuzzify the crisp value of this input into a fuzzy value, three special membership functions are applied, Fig. 4(a). By applying this input in the FG-RRT algorithm, the search tree growth direction is towards the target point in the complex configuration spaces.

3.2. Second input

This input in FIS is a metaheuristic function, inspired from the bees' behavior. The mathematical model of bees' behavior for food search in nature is presented as (4).

$$fit(q_{Parent}) = \begin{cases} \frac{1}{1+f(q_{Parent})}, & f(q_{Parent}) >= 0, \\ 1+abs(f(q_{Parent})), & f(q_{Parent}) < 0, \end{cases}$$
$$p_{q_{Parent}} = \frac{fit(q_{Parent})}{\sum_{K=1}^{SN} fit(q_{Parent})}, \tag{4}$$

Algorithm 1 evaluating competency of the selected parent vertex in the FG-RRT

1: $M \leftarrow$ *Determining maximum iteration for* q_{Parent}
2: *selection*
3: $T_C \leftarrow q_{Parent}$ *has a Low competency*
4: **while** $j \leq M$ *and* $T_C ==$ *Low* **do**
5: $d_E(q_{Parent}) \leftarrow$ *Calculated Euclidean function for* q_{Parent} *by equation* (3)
6: $fit(q_{Parent}) \leftarrow$ *Calculated metaheuristic function for* q_{Parent} *by equation* (4)
7: $F_{d_E} \leftarrow$ *Fuzzify* $d_E(q_{Parent})$ *through the membership functions depicted in Fig. 4(a).*
8: $F_{fit(q_{Parent})} \leftarrow$ *fuzzify* $fit(q_{Parent})$ *through the membership functions depicted in Fig. 4(b).*
9: **For** $i = 1:9$
10: $F_{rule}(i) \leftarrow$ *fulfillment degree of rule No. (i) through the min-operator*$(F_{d_E}, F_{fit(q_{Parent})})$
11: **EndFor**
12: $F_{competency} \leftarrow$ *calculated aggregation of all rules through the max-operator*$(F_{rule}(i))$
13: $T_c \leftarrow Update(T_c(F_{competency}))$ *by Table 2*
14: **EndWhile**
15: $C_{competency} \leftarrow$ *crisp value of*$(F_{competency})$
16: *by Centroid of Area (CoA) method*

Table 2. Comparison the features of conventional RRT, greedy RRT and FG-RRT.

competency of the selected parent vertex		First input $d_E(q_{Parent})$		
		Small	Average	Large
Second input $fit(q_{Parent})$	Excellent	High (1)	High (2)	Middle (3)
	Good	High (4)	Middle (5)	Low (6)
	Poor	Middle (7)	Low (8)	Low (9)

where $fit(q_{Parent})$ is the random offspring vertices count, generated by the selected parent vertex (q_{Parent}) which itself or its edge is located in the obstacle space. In the other words,$fit(q_{Parent})$ is the bee's unsuccessful efforts count in reaching the target point or determining the appropriate location in the configuration space which contains the adequate nectar. The (q_{Parent}) is the probability of choosing (q_{Parent}) as a parent vertex among all vertices of the search tree.

The functionality of each one of the vertices in this FG-RRT is similar to that of a bee in the nature and they collaborate with one another to find the target point or localities in the configuration space which is rich in nectar (food). For this purpose, each vertex of search tree becomes a simple computational component where through cooperation with other vertices can design the initial path. Metaheuristic function presented by (4) is applied by ver-

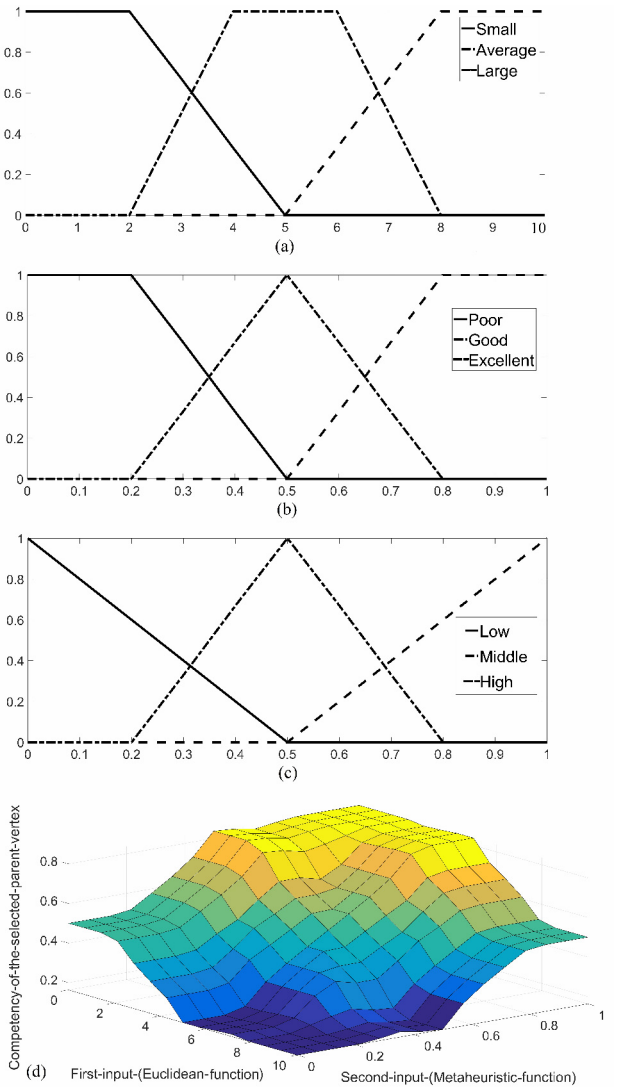

Fig. 4. Membership functions applied to fuzzify two inputs and one output are shown by (a), (b) and (c), respectively and rules surface for the FIS is shown by (d).

tices to exchange information and inform one another. By applying this input in the FG-RRT algorithm, the density of edges in search tree is controlled in the different places of configuration spaces. In order to fuzzify the crisp value of the second input, three special membership functions are applied, Fig. 4(b). The relation between the two inputs and one output is expressed by the nine linguistic rules tabulated in Table 2. For example, rule 1 in Table 2 indicates that: **If** the first input is small and the second input is excellent **Then** competency of the selected parent vertex is high. In other words, the selected parent vertex by the FG-RRT algorithm has a highest competency to become a parent vertex for generating a random offspring vertex, if $d_E(q_{Parent})$ in (3) is small and $fit(q_{Parent})$ in (4) is excellent after the fuzzification. The pseudo-code of that is

Algorithm 2 FG-RRT path planning algorithm

$q_{init} \leftarrow$ *initial vertex of search tree*
$q_{target} \leftarrow$ *target vertex of search tree*
$x \leftarrow$ *configuration space*
$N \leftarrow$ *Maximum iteration for path planning*
$\mathbf{T}_{edges} \leftarrow \{\varnothing\}$
$\mathbf{T}_{vertices} \leftarrow \{q_{init}, q_{target}\}$
$\mathbf{T}(\mathbf{T}_{edges}, \mathbf{T}_{Vertices}) \leftarrow$ *search tree*
$\mathbf{T}_C \leftarrow L^2$ *norm* (q_{new}, q_{target})
$\beta \leftarrow$ *threshold of* q_{target}
while $i \leq N$ *and* $T_c \geq \beta$ **do**
    $q_{parent} \leftarrow$ *Select* (q_{parent}) *randomly from the* $\mathbf{T}_{vertices}$
    $q_{parent} \leftarrow$ *Evaluating competency of* (q_{parent}) *through the algorithm 1*
    $q_{rand} \leftarrow$ *sampling function* (x_{free}, q_{parent})
    $[q_{new}, \gamma] \leftarrow$ *branch function* (q_{parent}, q_{rand})
    if *Output of the collision detection function* $(q_{new}, q_{rand}, \gamma)$*located in* x_{free} **then**
        $\mathbf{T}_{edges} \leftarrow \mathbf{T}_{edges} \cup \gamma$
        $\mathbf{T}_{vertices} \leftarrow \mathbf{T}_{vertices} \cup q_{new}$
        $\mathbf{T} \leftarrow Update(\mathbf{T})$
    EndIf
    $T_c \leftarrow Update(T_c)$
EndWhile

expressed in Algorithm 1.

3.3. Fuzzy inference system

The Mamdani FIS describes the relation between the two inputs and one output through the nine rules. The fuzzy rules are stated through the following structure; Rule i: **If** $d_E(q_{parent})$ is $MF^i{}_{d_E}$ and $fit(q_{Parent})$ is $MF^i{}_{fit(q_{Parent})}$ **Then** $comp_{q_{Parent}}$ is $MF^i{}_{comp(q_{Parent})}$,$i = 1, ..., 9$, where the first input ($d_E(q_{parent})$: quantity of the selected parent vertex) is computed through (3) and fuzzified by the three membership functions ($MF^i{}_{d_E}$), Fig. 4(a). The second input ($fit(q_{Parent})$: quality of the selected parent vertex) is computed through (4) and fuzzified by the three membership functions($MF^i{}_{fit(q_{Parent})}$), Fig. 4(b). The output ($comp_{q_{Parent}}$) is the competency of the selected parent vertex for generating a random offspring vertex and fuzzified by the three membership functions ($MF^i{}_{comp(q_{Parent})}$), Fig. 4(c).

The nine rules of FIS through the surface method are shown in Fig. 4(d). The fulfillment degree of each one of the rules and its implication is calculated through min-operator. The aggregation of the priorities of FIS is obtained through the max-operator. The crisp value of FIS output is obtained through the Centroid of Area (CoA) method. The pseudo-code of the FG-RRT is expressed in Algorithm 2 and its flowchart is shown in Fig. 5. Parameters β, N and M in Algorithms 2 and 1 are designed according to the path planning requirements and complexity of the configuration space, respectively.

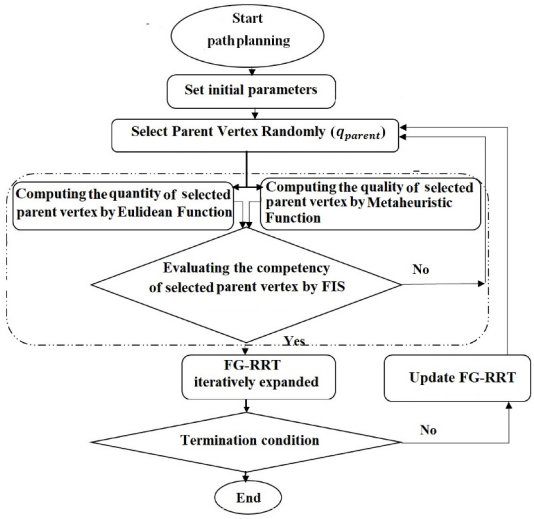

Fig. 5. FG-RRT path planning algorithm flowchart.

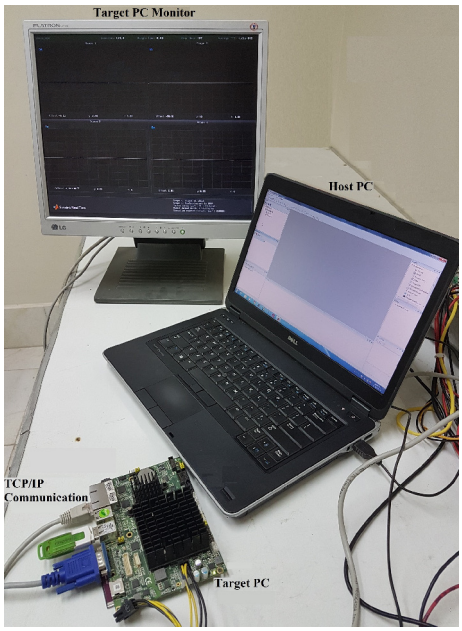

Fig. 6. PIL test setup.

3.4. The implementation of the proposed FG-RRT

Path planning ability of this algorithm is evaluated through PIL test which is an intermediate test usually run before Hardware-in-the-Loop (HIL) test and after Software-in-the-Loop (SIL) test [40]. The proposed FG-RRT with FIS is verified and debugged, by the means of the PIL test without actual AUxV system involvement. For this purpose, execution codes here are generated through the xPC Target builder and then implemented on a separate SBC through TCP/IP communication protocol according to maximum 10 Mbit/sec data transfer rate.

Table 3. Comparison of FG-RRT, greedy RRT and conventional RRT results in four test-cases with different complexity.

Path Planning Algorithm	Configuration space	Number of total vertices	Number of valid vertices	Initial path cost	Optimized path cost	Runtime
Conventional RRT	CCS	8849	6254	22	20.98	19.3
	NPCS	Max	–	–	–	INF
	MCS	Max	–	–	–	INF
	COCS	Max	–	–	–	INF
Greedy RRT	CCS	983	295	25	21.4	1.87
	NPCS	1382	493	42	37.31	3.48
	MCS	Max	–	–	–	INF
	COCS	Max	–	–	–	INF
FG-RRT	CCS	151	72	24	20.97	0.6
	NPCS	1138	687	38	33.42	2.53
	MCS	6442	3399	67	56.86	41
	COCS	6974	3305	56	45.88	43.5

The setup of the PIL test is shown in Fig. 6, consisting of three major parts of: host PC, target PC and xPC Target. The host PC is a laptop with an Intel Core i74700HQ @ 2.4GHz and 8GB RAM, the target PC is an Axiomtek SBC 84710 and xPC Target is a cost-effective method for running the PIL test. The FG-RRT path planning algorithm and configuration space are modeled and built in the host PC and then the FG-RRT kernel code is transferred to the target PC for execution.

4. TEST-CASES

To evaluate the path planning ability of FG-RRT algorithm four test-cases are designed through MATLAB R2016a with different complexity, Fig. 7(a)-(d). The first test-case is a cluttered configuration space (CCS) with 70 static obstacles in, Fig. 7(a); the second test-case is a configuration space with few narrow passages (NPCS) and 102 static obstacles in, Fig. 7(b); the third test-case is a maze-shaped configuration space (MCS) with 111 static obstacles in, Fig. 7(c) and the fourth test-case is a configuration space with combination of convex and concave obstacles, narrow passages, maze-like space (COCS) and 120 static obstacles in, Fig. 7(d). The results of applying the FG-RRT path planning algorithm with its FIS in these four configuration spaces are illustrated in, Figs. 8-11 and the results, here are tabulated in Table 3 for comparison purposes. The places with less nectar are shown in yellow rectangular, Figs. 8-11. These rectangular are obtained through the second input of the FIS in the FG-RRT which detects inappropriate places for exploring through the search tree; thus, FG-RRT prevents the tree edges' density increase in these places. This prevention allows search tree to have opportunity to expand and explore in the appropriate non-yellow places. In this process the vertices and edges count of the search tree do not increase, therefore the computational complexity and runtime remains constant. The valid edges, invalid edges, initial path and the final path of the search tree are shown in back solid lines, red dash lines, thick black line and thick red line, respectively. The triangular inequality optimization method is adopted to the initial path in order to plan the final path. The following results are summarized from Table 3 and Figs. 8-11:

1) The conventional RRT algorithm is able to plan the path only in the configuration space, Fig. 7(a), while in other three configurations with respect to the maximum number of vertices (15000 vertices) it is not able to plan the same.
2) According to the findings here, the greedy RRT path planning algorithm is able to plan the path in Figs. 7(a)-(b), while in the other two configurations with respect to higher degree of complexity it is not able to plan an initial path regarding the maximum number of vertices.
3) According to results obtained from the PIL test for the first test-case which are tabulated in Table 3 and shown in Fig. 8, the conventional RRT plans the initial path with 8849 vertices in19.3 seconds, the greedy RRT plans the initial path with 983 vertices in 1.87 seconds and the FG-RRT algorithm plan the initial path with 151 vertices in 0.6 seconds. The FGRRT algorithm is 32 times faster than conventional RRT and 3 times faster than greedy RRT. It is notable that, FG-RRT requires about 98% fewer vertices in comparison with conventional RRT and about 85% fewer vertices in comparison with greedy RRT for path planning.
4) According to results of the PIL test for the second test-case which are tabulated in Table 3 and shown in Fig. 9, the FG-RRT and greedy RRT algorithms plan an initial path through 1138 vertices in 2.53 seconds and 1382 vertices in 3.48 seconds, respectively, while the conventional RRT is not able to plan an initial path

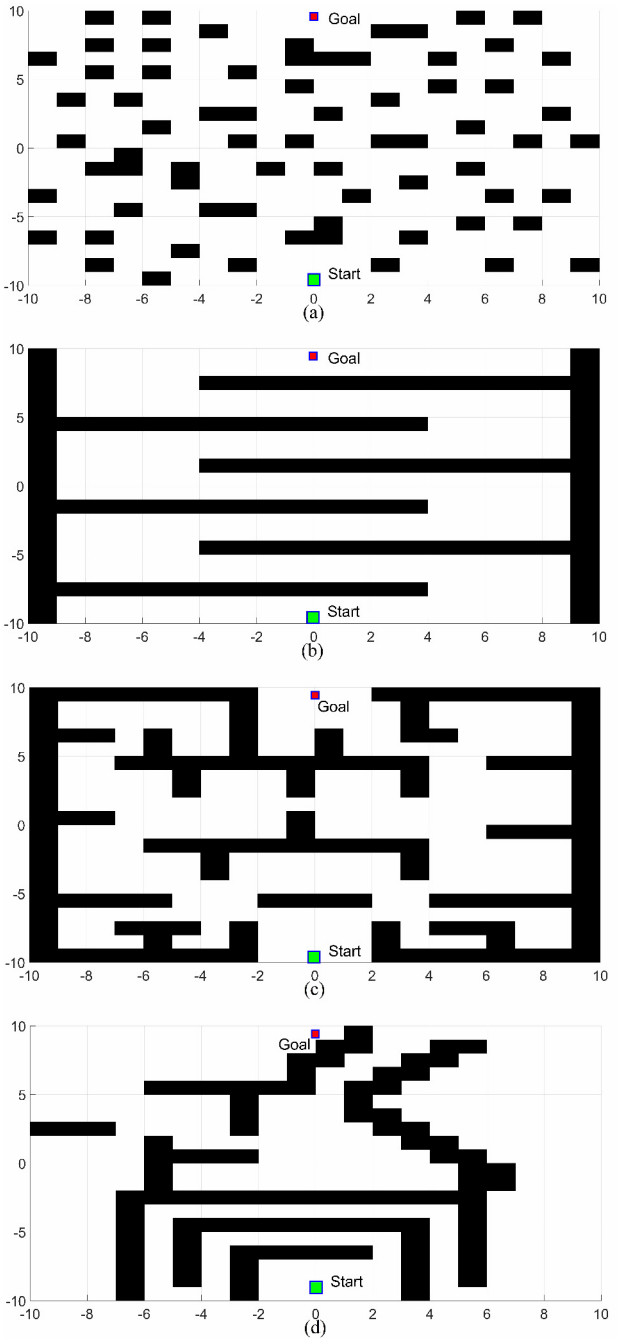

Fig. 7. A cluttered space with 70 static obstacles (a), Few narrow passages with 102 static obstacles (b), A maze-shaped space with 111 static obstacles (c) and A combination of convex obstacle, narrow passage and maze-like space with 120 static obstacles (d).

by the maximum number of vertices and the required runtime for path planning is infinite (INF).

5) In the third and fourth test-cases only FG-RRT algorithm is able to plan an initial path, while the greedy RRT and conventional RRT algorithm are not able to plan an initial path by the maximum number of vertices, Figs. 10-11.

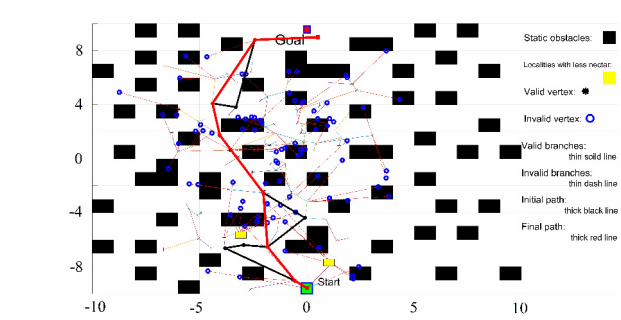

Fig. 8. Path planning in a cluttered space through the FG-RRT algorithm.

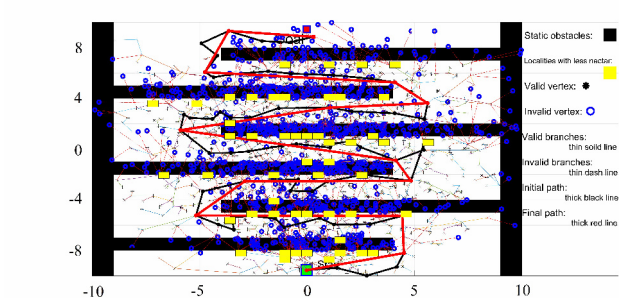

Fig. 9. Path planning in a configuration space with few narrow passages through the FG-RRT algorithm.

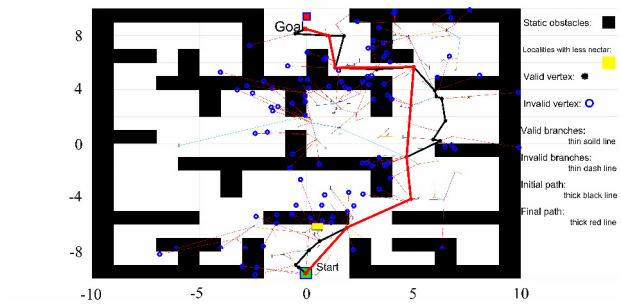

Fig. 10. Path planning in a maze-like space through the FG-RRT algorithm.

5. CONCLUSIONS

The proposed FG-RRT algorithm consist of a FIS with two inputs, one output and nine fuzzy rules to reduce the computational complexity and increase the runtime speed of path planning in complex configuration spaces. The first input is a Euclidean function applied in evaluating the quantity of the selected parent vertex and the second one is a metaheuristic function applied in evaluating the quality of selected parent vertex. By applying the first input in the FG-RRT the search tree growth direction is able to expand towards the target point and by applying the second

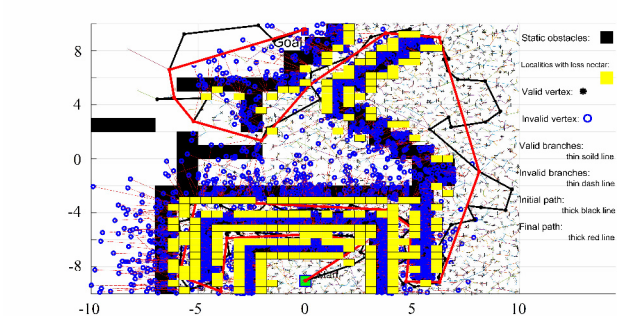

Fig. 11. Path planning in a complex configuration space with different combination of environmental constraints through the FG-RRT algorithm.

one the density of edges in the search tree becomes controllable in exploring appropriate places of configuration space. Output of FIS in the FG-RRT indicates the competency of selected parent vertex for generating a random offspring vertex. FG-RRT is implemented on Axiomtek SBC 84710 through the xPC Target builder and then evaluated through PIL tests. For this purpose, four test-cases are designed with different complexity. The results from the first and second test-case indicate that FG-RRT plans an initial path through fewer vertices in comparison with the conventional and greedy RRT; in the third and fourth test-cases the FG-RRT is only able to plan an initial path in the maximum number of vertices.

REFERENCES

[1] A. Finn and S. Scheding, *Developments and Challenges for Autonomous Unmanned Vehicles*, Springer, Berlin, 2012.

[2] C. C. Insaurralde, *Intelligent Autonomy for Unmanned Marine Vehicles*, Springer, Switzerland, 2015.

[3] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "Heuristic approaches in robot path planning: a survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13-28, 2016.

[4] R. Grabowski, "Big picture for autonomy research in DoD," *Soft and Secure Systems and Software and SW Symposium*, 2015.

[5] T. L. Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560-570, 1979.

[6] S. Tang, W. Khaksar, N. Ismail, and M. Ariffin, "A review on robot motion planning approaches," *Pertanika Journal of Science and Technology*, vol. 20, no. 1, pp. 15-29, 2012.

[7] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning - a chronological review," *World Academy of Science, Engineering and Technology*, vol. 29, no. 1, pp. 101-106, 2007.

[8] Y. Quinonez, F. Barrera, I. Bugueno, and J. B. Calfa, "Simulation and path planning for quadcopter obstacle avoidance in indoor environments using the ROS framework," *Proc. of the International Conf. Software Process Improvement*, pp. 295-304, 2017.

[9] M. Candeloro, A. M. Lekkas, and A. J. Sorensen, "A voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Engineering Practice*, vol. 61, pp. 41-54, 2017.

[10] E. Masehian and M. R. A. Naseri , "A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning," *Journal of Field Robotics*, vol. 21, no. 6, pp. 275-300, 2004.

[11] R. Gonzalez, M. Kloetzer, and C. Mahulea, "Comparative study of trajectories resulted from cell decomposition path planning approaches," *Proc. of the 21st International Conf. System Theory, Control and Computing*, pp. 49-54, 2017.

[12] Y. Chen, G. Luo, Y. Mei, J. Yu, and X. Su, "UAV path planning using artificial potential field method updated by optimal control theory," *International Journal of Systems Science*, vol. 47, no. 6, pp. 1407-1420, 2014.

[13] J. T. Schwartz, and M. Sharir, "On the piano movers' problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on Pure and Applied Mathematics*, vol. 36, no. 3, pp. 345-398, 1983.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846-894, 2011.

[15] L. E Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566-580, 1996.

[16] S. M. Lavalle, "Rapidly-exploring random trees: a new tool for path planning," *Citeseer*, 1998.

[17] K. Yang, Y. Kang, and S. Sukkarieh, "Adaptive nonlinear model predictive path-following control for a fixed-wing unmanned aerial vehicl," *International Journal of Control, Automation and Systems*, vol. 11, no. 1, pp. 65-74, 2013.

[18] M. C. Kim and J .B. Song, "Informed RRT* with improved converging rate by adopting wrapping procedure," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 53-60, 2017.

[19] I. Noreen, A. Khan, and Z. Habib, "A comparison of RRT, RRT* and RRT*-smart path planning algorithms," *International Journal of Computer Science and Network Security*, vol. 16, no. 10, pp. 20-27, 2016.

[20] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. L. Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 2537-2542, 2012.

[21] J. J. Kuffner and S. M. LaValle, "RRT-connect: an efficient approach to single-query path planning," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 995-1001, 2000.

[22] D. Schneider, E. Schomerr, and N. Wolpert, "Completely randomized RRT-connect: a case study on 3D rigid body motion planning," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 2944-2950, 2015.

[23] D. J. Webb and J. V. D. Berg, "Kinodynamic RRT*: asymptotically optimal motion planning for robots with linear dynamics," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 5054-5061, 2013.

[24] R. C. Luo and C. Huang, "Anytime dynamic exploring rapid random tree approach in higher dimension search space for non-holonomic robotics," *Proc. of the International Conf. Advanced Robotics and Intelligent Systems*, pp. 1-6, 2016.

[25] Y. Dong, E. Camci, and E. Kayacan, "Faster RRT-based nonholonomic path planning in 2D building environments using skeleton-constrained path biasing," *Journal of Intelligent and Robotic Systems*, vol. 89, pp. 387- 401, 2018.

[26] L. Palmieri, S. Koenig, and K. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 2775- 2781, 2016.

[27] F. Burget, M. Bennewitz, and W. Burgard, "BI2RRT*: An efcient sampling-based path planning framework for task-constrained mobile manipulation," *Proc. of the IEEE International Conf. Intelligent Robots and Systems*, pp. 3714-3721, 2016.

[28] D. Kim, J. Lee, and S. Yoon, "Cloud RRT*: Sampling cloud based RRT*," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 2519-2526, 2014.

[29] M. Elbanhawi, M. Simic, and R. Jazar, "Randomized bidirectional b-spline parameterization motion planning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 406-419, 2016.

[30] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, "Optimal path planning in cluttered environment using RRT*-AB," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 41-52, 2018.

[31] R. Cui, Y. Li, and W. Yan, "Mutual information based multi-AUV path planning for scalar field sampling using multidimensional RRT*," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 993-1004, 2016.

[32] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473-483, 2016.

[33] D. Devaurs, T. Simeon, and J. Cortes, "A multi-tree extension of the transition-based RRT: application to ordering-and-pathfinding problems in continuous cost spaces," *Proc. of the IEEE International Conf. Intelligent Robots and Systems*, pp. 2991-2996, 2014.

[34] W. Y. Shin, J. J. Shin, B. Kim, and K. Jeong, "Line segment selection method for fast path planning," *International Journal of Control, Automation and Systems*, vol. 15, no. 3, pp. 1322-1331, 2017.

[35] C. H. Kim and S. Sugano, "Closed loop trajectory optimization based on reverse time tree," *International Journal of Control, Automation and Systems*, vol. 14, no. 6, pp. 1404-1412, 2017.

[36] K. Yang, "Anytime synchronized-biased-greedy rapidly-exploring random tree path planning in two dimensional complex environments," *International Journal of Control, Automation and Systems*, vol. 9, no. 4, pp. 750-758, 2011.

[37] C. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1080-1090, 2015.

[38] L. Palmieri and K. Arras, "Distance metric learning for RRT-based motion planning with constant-time inference," *Proc. of the IEEE International Conf. Robotics and Automation*, pp. 637-643, 2015.

[39] S. Khanmohammadi and A. Mahdizadeh, "Density avoided sampling: an intelligent sampling technique for rapidly-exploring random trees," *Proc. of the Eighth International Conf. Hybrid Intelligent Systems*, pp. 672-677, 2008.

[40] S. Lee, H. Bang, and D. Lee, "Predictive ground collision avoidance system for UAV applications: PGCAS design for fixed-wing UAVs and processor in the loop simulation," *Proc. of the International Conf. Unmanned Aircraft Systems*, pp. 1287-1292, 2016.

Ehsan Taheri was born in Iran in 1984. He received his B.Sc. degree in electrical engineering and his M.S. degree in control engineering, in 2006 and 2008, respectively from the Islamic Azad University, Najafabad Branch and Malek Ashtar University of Technology. Currently, he is a Ph.D. candidate in the Malek Ashtar University of Technology. His research interests include Autonomy, Underwater Robots, Path planning, heuristic optimization, and Motion Control.

Mohammad Hossein Ferdowsi received his BSc and MSc degrees in electrical engineering from Sharif University of Technology and his Ph.D. degree in electrical engineering from University of Tehran, in 1977, 1980, and 2004, respectively. From 1985 to 1987, he was a lecturer at Sharif University of Technology, and since 1987, has been at Malek Ashtar University of Technology as a faculty member. His research interest is in multivariable and adaptive control systems, intelligent systems, and target tracking.

Mohammad Danesh received his B.Sc., M.Sc., and Ph.D. degrees in control engineering from the Isfahan University of Technology (IUT), Isfahan, Iran, in 1997, 1999, and 2007, respectively. He has been with the department of Mechanical Engineering, IUT, since 2007. His current research interests include robotics, intelligent systems, mechatronics, control of dynamical systems, and stability analysis.