RRT-A-BT* approach for optimal collision-free path planning for mobile robots

Abdelfetah Hentout[1], Abderraouf Maoudj[2], Djelloul Yahiaoui[3] and Mustapha Aouache[4]
[1,2,4]Centre de Développement des Technologies Avancées (CDTA)
BP 17, Baba Hassen, Algiers 16303, Algeria.
[3]Université Saad Dahleb de Blida (USDB), Département d'Informatique
BP 270, Route de Soumaa, Blida 09100, Algeria.
[1]ahentout@cdta.dz, [2]amaoudj@cdta.dz, [3]yahiaoui.djelloul01@gmail.com, [4]maouache@cdta.dz

Abstract: This paper deals with the problem of optimal collision-free path planning for mobile robots evolving inside indoor cluttered environments. Addressing this challenge, a hybrid approach is proposed combining *Rapidly-exploring Random Trees* (*RRT*), *A-Star* (*A**) and *Back-Tracking* (*BT*) algorithms (*RRT-A*-BT*). Thus, a vision system is used for a nearly-exact modeling of the environment through image processing. Moreover, each iteration of the basic *RRT* approach is guided by *A** algorithm while trying to take the shortest path linking the robot current position to $Target$. Incase of a blockage, *BT* algorithm is used to get out the robot from this situation. Finally, *Piecewise Cubic Hermite Interpolating Polynomial* (*PCHIP*) is used to smooth the generated optimal path. *RRT-A*-BT* approach is validated through different scenarios; obtained results are compared with previous works on same environments with same conditions. The results prove that *RRT-A*-BT* is better and faster than other algorithms of the literature, such as *Genetic Algorithms* and *Conventional RRT*, in terms of (*i*) *computation time*,(*ii*) *path length* and (*iii*) *transfer time*.

Keywords: Optimal path planning, Collision-free path, Indoor mobile robotics, *Rapidly-exploring Random Trees*, *A-Star*, *Back-Tracking*.

1. INTRODUCTION

Optimal collision-free path planning is an important challenge in mobile robotics research field. It consists of finding the best path moving the robot from its initial configuration ($Source$) to the imposed destination ($Target$) while avoiding possible obstacles [1]. These configurations must satisfy the specified performance constraints such as *transfer time*, *path length*, *energy*, etc. or a *combination of them* [2].
The approach described in this work is an improvement of that presented in [3], in which a collision-free path planning algorithm is proposed for mobile robots evolving in indoor workspaces cluttered with static obstacles. This algorithm is based on *Rapidly-exploring Random Trees* (*RRT*) and *Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)*.

Cite this article as:
Abdelfetah Hentout, Abderraouf Maoudj, Djelloul Yahiaoui and Mustapha Aouache, "*RRT-A*-BT* approach for optimal collision-free path planning for mobile robots", ", Algerian Journal of Signals and Systems, Vol. 4, Issue 2, December 2019.pp: 39-50.

However, *RRT-PCHIP* approach did not guarantee the optimality of found paths. This paper describes the development of a hybrid approach combining *RRT*, *A-Star* (*A**) and *Back-Tracking (BT)* algorithms, *RRT-A*-BT*, to tackling the considered problem. So far, to the best of the authors knowledge, no published study dealt with such a problem by hybridizing *RRT*, *A** and *BT* algorithms.
The contributions of this work can be listed as follows. First, the proposed approach deals with global path planning with a nearly-exact representation of the environment. Second, the mobile robot exploits a vision system to model its cluttered workspaces through image processing. Third, each iteration of the basic *RRT* approach is guided by *A** algorithm while trying to take the shortest path linking the robot current position to $Target$. Finally, a comparative study is carried out via several scenarios for a robot. It allows affirming that the proposed *RRT-A*-BT* approach is better and faster than other methods in the literature, such as *GA-PCHIP* [4] and *Conventional RRT* [5], especially in terms of (*i*) *computation time*, (*ii*) *path length* and (*iii*) *transfer time*.

The remainder of this paper is given as follows. Section 2 surveys previous work on optimal collision-free path planning for mobile robots evolving in indoor environments cluttered with static obstacles. Section 3 gives a short description of this problem. Section 4 describes the proposed approach based on *RRT*, *A** and *BT* algorithms. Section 5 presents, discusses and compares the main obtained results on the used experimental robotic system. Section 6 concludes the paper and presents future works.

2. Related works

Numerous studies and efforts have been made to tackling this problem resulting in a considerable number of solutions [6].

In the earlier days, mathematical models were developed to generate optimal paths for mobile robots such as *polynomial*, *cubic splines* and *Bezier curve* [7]. However, these algorithms were either ineffective, because of the significant computation cost; or imprecise, because of getting stuck in local minima [3].

After that, researchers proposed various solutions based on traditional techniques such as *Voronoï Diagram* (*VD*) [8], *Cell Decomposition* (*CD*) [9] and *Potential Field* (*PF*) [10]. Nevertheless, the accuracy of the generated paths highly depends on the grids size (the higher the size the lower the accuracy). Further, these techniques easily fall into traps in complex problems [2].

In the other class of approaches, one can find soft-computing techniques such as *Fuzzy Logic* (*FL*) [11] and *Neural Network* (*NN*) [12]. These techniques exhibit good performances; on the other hand, the accuracy of the final paths highly depends on the rules given by the human expert (for *FL*), or on the training state (for *NN*).

The modern techniques are *Nature-Inspired Meta-Heuristic* (*NIMH*) algorithms such as artificial immune systems [13] and heuristic optimization algorithms [14]. Additionally, several evolutionary and differential evolution approaches have been explored. Some authors applied *Bacterial Foraging Optimization* (*BFO*); here, obstacles are enveloped by circles, and robot is approximated by a square [15]. However, the generation of a safe path required many seconds and depends on the number of obstacles, better than standard *BFO* and *Particle Swarm Optimization* (*PSO*) algorithms used by other authors [16] [17].

As stated in [7], hybrid techniques give better performances for optimal collision-free path planning problem. The authors in [4] combined *GA* with *FL* control using a vision system in cluttered workspaces. *PSO* has also been combined with other algorithms such as *Gravitational Search* (*GS*) [18], *Artificial Potential Field* (*APF*) [19], *Modified Frequency Bat* (*MFB*) [20], etc. Other researchers investigated the use of hybrid *Takagi-Sugeno FL* model and *Simulated Annealing* (*SA*) algorithms [21]. The carried experiments considered simple environments and showed effective navigation [22]. Saraswathi et al. [23] combined two *NIMH* algorithms; namely *Cuckoo-Search* (*CS*) [24] and *Bat Algorithm* (*BA*) [25]. This hybrid method took less time to reach $Target$ as compared to individual algorithms.

However, all the aforementioned approaches suffer of many drawbacks such as high computation cost, complexity in large workspaces, low accuracy of evolution environments depending on the size of the grids, the *FL* rules elaborated by the human expert, the training state of the *NN*, trapping in local minima, high sensitivity to search space size and data representation scheme, etc. [26] [27].

To deal with these issues, *Sampling-Based Planners* (*SBP*) have been proposed. These approaches are proved to be computationally efficient and are effective standard for high-dimension motion planning problems [28]. Among *SBP* algorithms, one can find *Probabilistic Roadmap Methods* (*PRM*) [29] [30] and *RRT* [31]. The first class was successfully utilized in many research works on high-dimension workspaces [32]. However, *PRM* are ineffective when obstacles geometry is unknown a priori [33]. To outdo this inefficiency, *RRT* approaches have been proposed; further, several variants enhancing *Standard RRT* have been explored [34]. The first variant, *RRT-connect*, checks if traveling straight in the line of expanded node reaches $Target$ [35]; in this case, the algorithm terminates. The second variant, *Bidirectional RRT*, grows two trees [36] (one is rooted at $Source$ and the other is rooted at $Target$); the algorithm ends when the two trees meet. The last variant, *RRT**, combines the basic *RRT* with *Rapidly-exploring Random Graph* (*RRG*) [37].

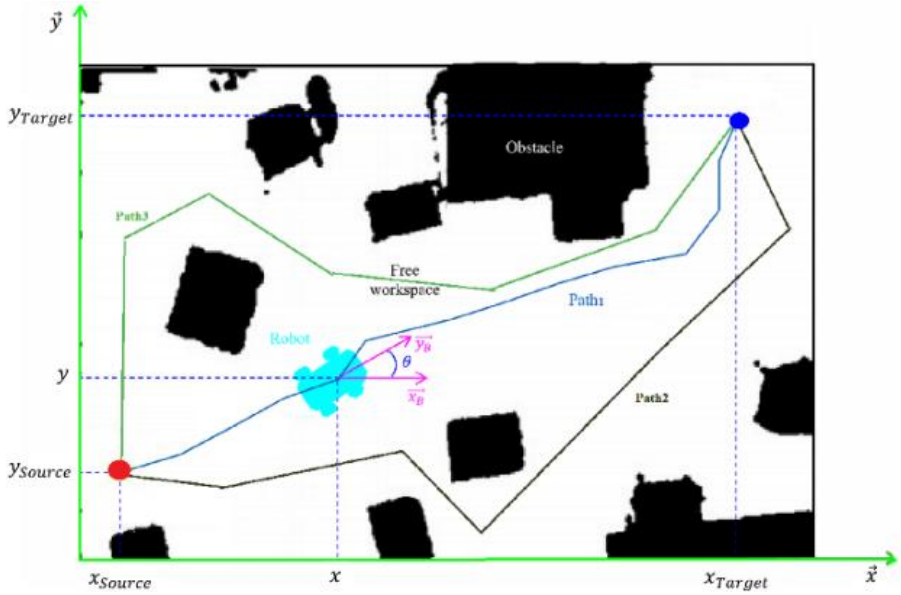


Fig. 1. Optimal collision-free path planning for mobile robots in cluttered environments. The white zone represents the free workspace; the black zones are the obstacles present in the workspace.

3. OPTIMAL COLLISION-FREE PATH PLANNING PROBLEM

This work describes an optimal collision-free path planning approach for mobile robots evolving in indoor environments cluttered with static obstacles. The found path links the initial configuration $Source(x, y)$ to the imposed final configuration $Target(x, y)$; neither $Source$ nor $Target$ positions are assumed to overlap with obstacles.

Table 1 gives the format of the generated path connecting $Source$ to $Target$ without collision. The feasible path consists of a set of nn nodes and $ns = nn - 1$ segments in form of (x, y) vectors.

Table 1: Format of the generated path connecting $Source$ to $Target$.

$Source$	2nd node	...	nsth node	$Target$
x_{Source}	x_2	...	x_{ns}	x_{Target}
y_{Source}	y_2	...	y_{ns}	y_{Target}

The path planning approaches for mobile robots are based on their kinematic model given by (1) [38]where:

- v: linear velocity of the robot,
- ω: angular velocity of the mobile robot,
- θ: orientation of the robot (on $\vec{x}$ axis).

$$\begin{cases} \dot{x} = v\,cos\theta \\ \dot{y} = v\,sin\theta \\ \dot{\theta} = \omega \end{cases} \quad (1)$$

Generally, the considered evolution workspaces may be complex. Fig. 1 describes a mobile robot evolving in a cluttered workspace with exact representation of many obstacles [3] (red disk represents $Source$; blue disk represents $Target$). The figure shows three possible paths (green, blue and black) leading the robot from $Source$ to $Target$. It is clear that the shortest one is the blue path (i.e., $Path_1$). The different constraints that must be satisfied during the movement of the robot ($t \in [0, T_{Transfer}]$) are adapted from [39] as follows:

$$minimize \begin{cases} T_{Transfer} \\ l \end{cases} \quad (2)$$

under the following constraints(where $q = (x, y)$ is the robot state):

$$q(t = 0) = Source(x, y) \quad (3)$$

$$q(t = T_{Transfer}) = Target(x, y) \quad (4)$$

$$v(t) \in [v_{min}, v_{max}], t \in [0, T_{Transfer}] \quad (5)$$

$$\omega(t) \in [\omega_{min}, \omega_{max}], t \in [0, T_{Transfer}] \quad (6)$$

$$l = \sum_{k=1}^{k=ns} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \quad (7)$$

$$Collision(Robot, Obstacle_i) = \emptyset, i \in \{1 \dots no\} \quad (8)$$

where:

- $T_{Transfer}$: final transfer time,
- l: length of the generated path,
- v_{min}, v_{max}: minimum and maximum linear velocities of the mobile robot,
- $\omega_{min}, \omega_{max}$: minimum and maximum angular velocities of the mobile robot,
- ns: number of segments of the path,
- no: number of obstacles in the workspace,

- $Collision$: a *Boolean* function that verifies if the robot is in collision with an obstacle $Obstacle_i, i \in \{1 \dots no\}$,
- $(x_k, y_k), (x_{k+1}, y_{k+1})$: coordinates of first and last node of segment k (Table 1).

4. PROPOSED RRT-A*-BT APPROACH

The proposed algorithm, *RRT-A*-BT*, is a hybridization between *RRT*, *A** and *BT*. This path planning and optimization approach is described in what follows.

4.1. Modeling of the cluttered environment

The mobile robots evolve in an environment cluttered with several obstacles. The locations of existing obstacles are completely or partially known. In our approach, a nearly-exact representation of this workspace is proposed while exploiting a vision system through images processing as described by the diagram of Fig. 2. The modeling of the evolution environment consists of obtaining a binary safe map where the robot can safely move from $Source$ to $Target$.

The various performed operations to model the environment can be outlined as follows. First, the *RGB* image of the workspace (Fig. 6(a)) is binarized as described by Fig. 6(b). The threshold for height to extract obstacles highly depends on the accuracy of the utilized vision sensor (*Kinect*); it is empirically fixed as $threshold$ = 100mm [4]. Second, the approximation with rectangles of existing obstacles is calculated; the result is shown by Fig. 6(c). Finally, to successfully move without collisions, and due to sensors uncertainties, the robot should keep a safe distance with obstacles to avoid collisions. For this aim, a safe region has been added to the present obstacles of the environment. The radius for this safety region is empirically fixed to 20$pixels$; $20pixels \approx 20 \times 6.34mm/pixel \approx 126.80mm$ (see subsection 5.1).

The evolution environment is therefore divided into three different zones: (*i*) *safe area(free workspace*), (*ii*) *high-risk area* (*vicinity of obstacles*) and (*iii*) *forbidden area* (*obstacles*). Consequently,the mobile robot is considered as a mass point moving inside this workspace with different speeds.

4.2. Collision-free path planning using RRT

RRT approach is proposed as a data structure and a sampling algorithm designed for efficient and fast search. It has been used for high-dimension workspaces in path planning problems with state constraints (obstacles, etc.) [40].

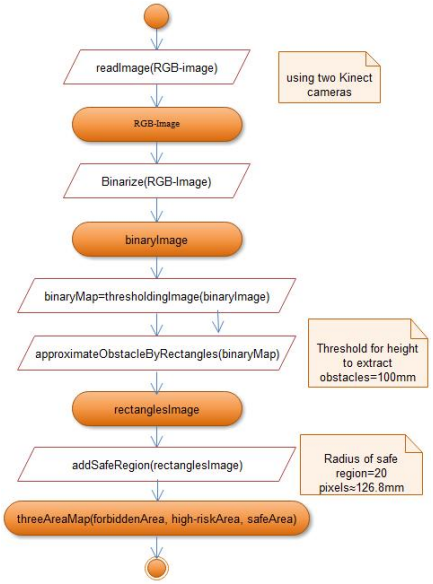


Fig. 2. Environment modeling diagram by images processing.

This approach progressively builds a tree G using random sampling towardunexplored regions of the environment. Gstarts from$Source$, as the root, and expands to find a path toward $Target$[35]. In every iteration, the approach generates a random node in the configuration space (*C-Space*), searches for the nearest node in G and extends this node to the random sample by apredefined step size [41]. The algorithm stops when it reaches$Target$ (in best case) [42] or one of the tree leaves reaches$Target$ region (in most cases).

The main advantages of *RRT* are simplicity (implementation is straightforward and only needs simple computations), fast convergence and expansion toward unexplored regions of *C-Space*, and probabilistically completeness [43] [44]. Nevertheless, *RRT* is sub-optimal [45] which may be global or local[42]. Global optimality indicates the strategy to avoid obstacles (go from left/right or in-between obstacles);whereas, local optimality indicates distances to maintain from obstacles.

At the beginning, the tree G is empty; $Source$ and $Target$ positions are introduced. Further, the parameters MAX_ITER, $STEP$ and $DIST$ are empirically set up as follows [36]:

- MAX_ITER: maximum of iterations; in this work, MAX_ITER = 10000,

- $STEP$: maximum length of the segment connecting two successive nodes; in this work, $STEP = 40\,pixels$,
- $DIST$: radius of the search region near $Target$; in this work, $DIST = 20\,pixels$.

The purpose is to create a random G connecting $Source$to$Target$. As Fig. 4 shows, the condition $region(New, Target)$ verifies if the new node New belongs to a region centered in $Target$; this means $\|New - Target \leq DIST$ (instead of testing if $New \equiv Target$ which would take much more time) [43].

While the above condition is not satisfied, the following process is repeated. First, a collision-freenode,$Rand$, is generated randomly in *C-Space* ($randomFreeNode(C - Space, MAX\_RAND)$); in this work $MAX\_RAND = 50$. After that, the nearest neighbor of $Rand$, already in G, is searched by utilizing the *Euclidean* distance calculated by (9) between New and $Rand$. Next, the node $Near$ is extended in the direction of $Target$ by $STEP$ (segment length between two successive nodes must not exceed predefined $STEP$ size). The expansion is interrupted when the node New is in collision with obstacles or it falls outside the boundary limits (physical limits of *C-Space*); in this case,$extendNode(Near, STEP)$ returns $NULL$. The generated path is found by following the predecessors of the last added node New toward $Source$ [43].

$$EuclDist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (9)$$

where:

- (x_1, y_1): Cartesian coordinates of $Node_1$,
- (x_2, y_2): Cartesian coordinates of $Node_2$.

4.3. A* algorithm

*A** algorithm, an extension of *Dijkstra* algorithm, is one of the best known path planning algorithms which can be applied on metric or topological *C-Spaces* [46]. *A** searches for a path in a graph between known initial node ($Source$) and known final node ($Target$).

The inputs of *A** algorithm are $C - Space$, $Source$, $Target$ and two lists of nodes that are initially empty (*Open list* and *Close list*). *Open list* contains all the paths that are still feasible; *Close list* contains all the studied solutions (good and bad). Fig. 3 summarizes the principle of this algorithm.

Each iteration, *A** tries to take the shortest path to go from $Source$ toward $Target$. To determine if a node is likely to be part of the solution, its quality must be quantified. Therefore, *A** uses a combination of heuristic searching and searching based on the shortest path; a value $f(Node)$ given by (10) is attached to each adjacent node of the current node [46] where:

- $h(Node)$: heuristic distance of the current $Node$ toward $Target$,
- $g(Node)$: length of the path from $Source$ to $Target$ through $Node$.

$$f(Node) = h(Node) + g(Node) \quad (10)$$

The node with the lowest value (i.e., $bestQuality$) is selected as the next one in the sequence [47] (Fig. 3):

- **Step 1**: from $Source$, look at the neighbor nodes; for example, one can go right, left backward or forward. $Source$ is inserted in *Close list* ($add(closeList, Source)$),
- **Step 2**: for each visited neighbor node ($currentNode$):
  - if $currentNode$ is an obstacle ($outsideSafeArea(currentNode) = true$), move on; this means that this node will not be inserted anywhere,
  - if $currentNode$ is already in *Close list* ($exist(closeList, currentNode) = true$), move on; this means that this path has been studied.
  - if $currentNode$ is in *Open list*, its quality will be verified. If its current quality is better ($f(currentNode) < bestQuality$), this path is shorter. *Open list* is thus updated ($add(openList, currentNode)$) with its parent link (with $currentNode$ that corresponds to the best path).
  - otherwise, this node is added to *Open list* with $currentNode$ as a parent.
- **Step 3**: look in *Open list* for the node with best quality:
  - if *Open list* is empty ($empty(openList) = true$), this means that there is no solution ($showMessage(No\ Solution!)$).
  - otherwise ($empty(openList) = false$), take the best node ($bestNode$), remove it from *Open list* ($openList = openList - bestNode$) and insert it in *Close list* ($add(closeList, bestNode)$).
- **Step 4**: $currentNode$=last inserted node.
- **Step 5**: if $bestNode = Target$ (i.e., $region(bestNode, Target) = true$), a

path has been found; otherwise, go to **Step 2**.

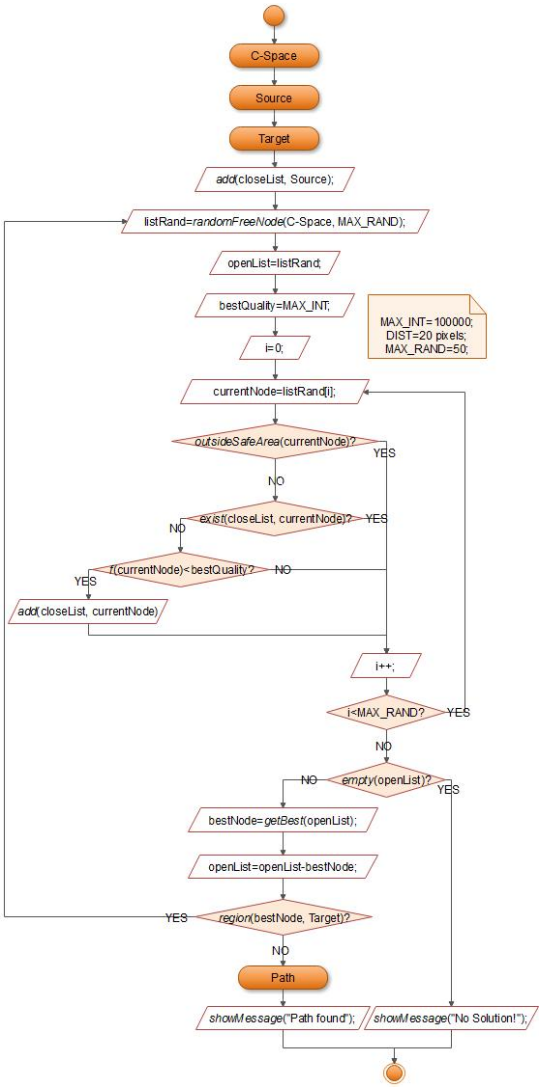


Fig. 3. Principle of *A** algorithm for optimal path searching.

4.4. Back-tracking algorithm

In some cases, a premature end may occur to the *RRT-A** path planning algorithm and fail to generate a feasible path. Indeed, after implementing this solution, a blocking problem has been encountered while dealing with workspaces containing *"U"* form obstacles (for example).

To be able to deal with such a blockage, the proposed planning approach should be able to track other possible paths. This pushed us to hybridize with another mechanism; we applied thus a modification to return back and exploit other feasible paths. To solve this, *BT*algorithm has been applied [48]. *BT* is a family of algorithms that consists of returning back on made decisions to get out of a blockage.

*BT*algorithm is used to start a new path from one of the previous nodes. In such a case, the algorithm recursively goes backward to the previous nodes. As a consequence, a nodes list is kept in case the algorithm must back-track to find a solution. As stated in [49], a mechanism must be included to determine the best starting node and to guide the robot from the current blocking node to $Target$. In the current version of the proposed approach, the starting backward node consists of its nearest one having the best quality value (see subsection 4.3). Each iteration, a new path is calculated starting from this new node. If this new path gets trapped again, the head of the path is reassigned to the previous node and another new path is regenerated again [50].

4.5. Reduction of the number of nodes

This approach is based on testing the collision of a given segment of the generated path with obstacles. The collision test is done between the first node of the current segment ($Path[i]$) and the second node of the next segment ($Path[i+2]$). In case of a collision ($collision(Path[i], Path[i+2]) = true$), the two edges are kept and added to the new path ($addEdge(NewPath, Path[i], Path[i+1])$ and $addEdge(NewPath, Path[i+1], Path[i+2])$).Otherwise ($collision(Path[i], Path[i+2]) = false$), a new segment is created between the two nodes and added to the new path ($addEdge(NewPath, Path[i], Path[i+2])$).This process is reiterated until reaching $Target$[3].

4.6. Smoothing the zigzag path

RRT-A-BT* stops when it finds the best path connecting without collision $Source$ to $Target$.

As Table 1 shows, this path consists of a set of nn nodes (ns connected segments) in form of (x, y) vectors. To be able to track the generated paths, the robot has to stop, to change its orientation and to restart frequently [2]. To overcome this, the ns discontinuous segments need to be smoothed. For this, many geometric curve-based smooth techniques have been proposed such as cubic splines [51], *polynomials* [52], *Bezier* curve [53] and *Non-Uniform Rational Basis Splines* (*NURBS*)

[54]. As in our previous works ([4] and [3]), this paperadopted *PCHIP* algorithm to smooth the resulted zigzag line path.

4.7. Proposed RRT-A*-BT

The proposed *RRT-A*-BT* solution aims to build an off-line optimal collision-free path connecting $Source$ to $Target$ for mobile robots evolving in indoor workspaces cluttered with static obstacles. This hybrid approach combines *RRT*, *A** and *BT* algorithms. Fig. 4 recapitulates the main steps of this approach.

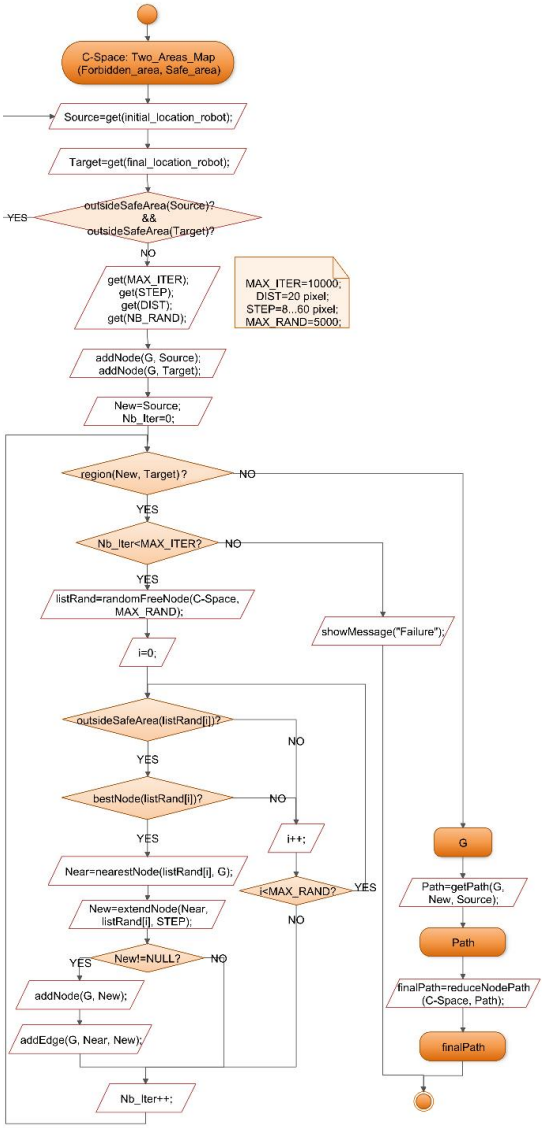


Fig. 4. Diagram of the proposed *RRT-A*-BT* path planning approach.

5. VALIDATION AND DISCUSSION

The proposed approach is validated through several scenarios on the available experimental robotic system. For the following figures, the *white zone* represents the*safe area* (*free workspace*), the *gray zone* represents the*high-risk area* (*near obstacles*), the *black zones* represent the*forbidden area* (*obstacles*), the *blue rectangles* represent the *approximation of obstacles by polygons* and the *red line* represents the *optimal path*.

5.1. Architecture of the robotic system

The experimental robotic system is composed of two distant sites with a wireless connection (Fig. 5) [55]:

- *Operator site (local site)*: an off-board host PC is used to send orders to the remote robot through wireless connection,
- *Robot site (Remote site)*: a rectangular mobile manipulator, *RobuTER/ULM*, is controlled by an on-board industrial PC.

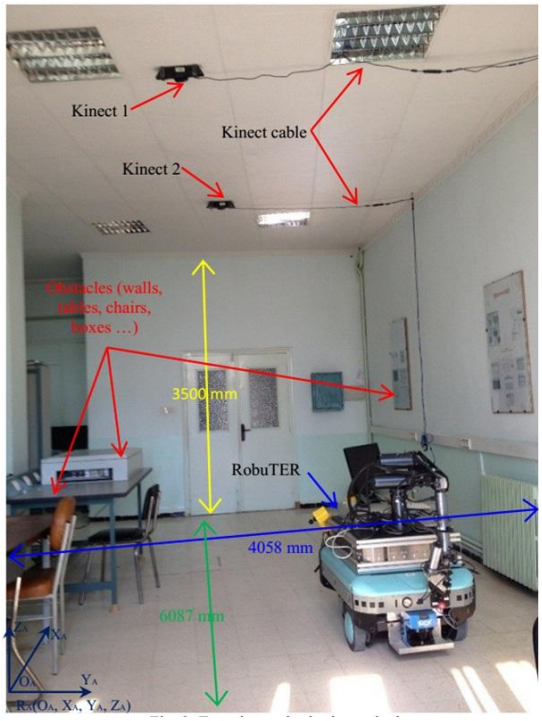


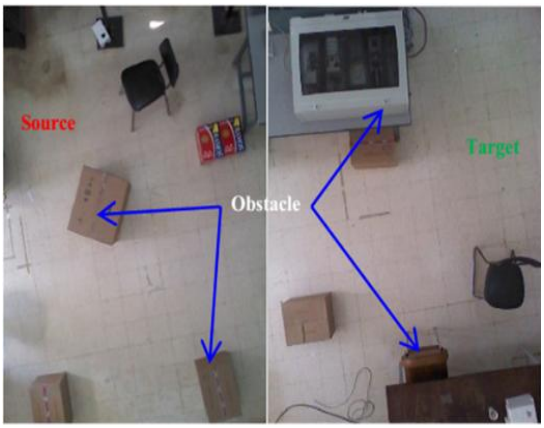
Fig. 5. Experimental robotic system test-bed [4].

The robot environment is perceived using two *Microsoft Kinect* cameras *V1*. The image stream contains color data in *RGB* format with a resolution of 640 × 480 and 30 *frames per second*. The cameras are fixed on the roof of the workroom at a $height$ = 3500mm (Fig. 5). Further, they are connected to the off-board PC to acquire and process images. The resolution of the pictures delivered by the vision system is 640 × 960 $pixels$ and covers a range of about 4058 × 6087mm

($1\ pixel \approx 6.34mm$). In this work, we only consider the *RobuTER* mobile base; the *ULM* manipulator is not utilized.

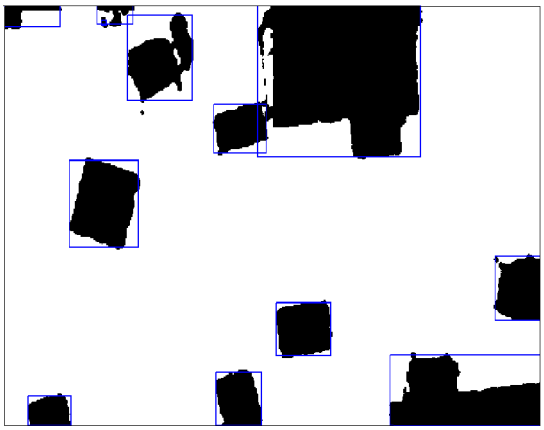
5.2. Validation workspaces

RRT-A-BT* selects the optimal path according to the considered criteria. It should be executed several times with the step size determined in [3] to minimize (*i*) *computation time*, (*ii*) *length of the generated path*, (*iii*) *number of segments* and (*iv*) *transfer time*. For this, the best step size is 40 $pixels$ corresponding to 253.60mm, approximately.

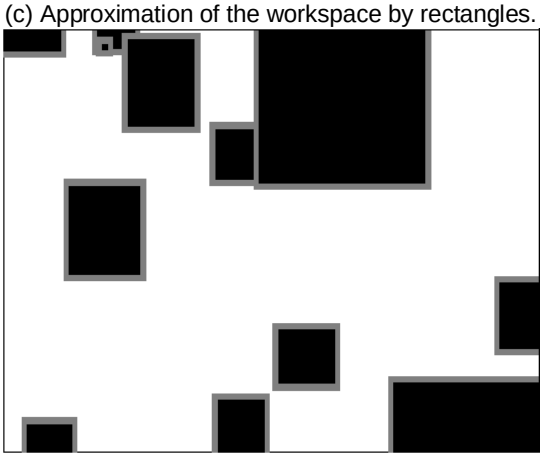


(a) Original workspace cluttered with static obstacles [4].

(b) Binary map of the workspace.



(c) Approximation of the workspace by rectangles.



(d) Binary safe map of the workspace.

Fig. 6. Evolution workspace of the robot [4].

5.3. Comparison with previous works

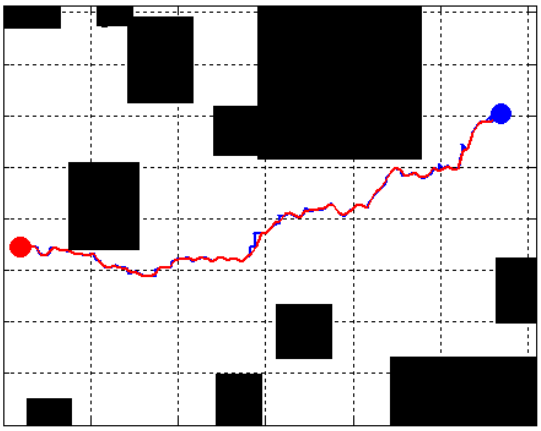
To compare the proposed *RRT-A*-BT* approach with other similar works, the same validation scenarios as considered in [4] (for *GA-PCHIP*) and [5] (for *ConventionalRRT*) are used.It must be noted that the initial and final orientations of the robot (θ_B) are not taken into account when dealing with the scenarios.

5.3.1. Comparison with *GA-PCHIP*

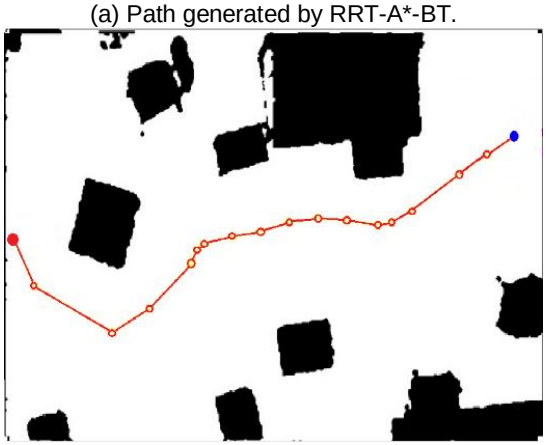
Fig. 6(a) shows the original workspace cluttered with static obstacles; whereas, Fig. 6(d) shows the binary safe workspace and its approximation with polygons.

The mobile robot, *RobuTER*, has to move from $Source(x, y)$ = (74mm, 1953mm) toward$Target(x, y)$ = (5501mm, 1308mm) with a maximum linear speed of v_{max} = 150mm/s. The boundary velocities are fixed to null ($v(t$ = 0) = 0) and ($v(t = T_{Transfer})$ = 0).

Fig. 7(a) and 7(b) represent the paths obtained by *RRT-A*-BT* and *GA-PCHIP*[4], respectively.



(a) Path generated by RRT-A*-BT.



(b) Path generated by *GA-PCHIP*.

Fig. 7. *RRT-A*-BT* versus *GA-PCHIP* [4].

Table 2 compares the obtained results. It shows that *GA-PCHIP* [4] generates a longer path from $Source$ to $Target$ in much more time compared with the proposed *RRT-A*-BT*.

Table 2: Comparison between the proposed *RRT-A*-BT* and *GA-PCHIP* [4]

Parameters	RRT-A*-BT	GA-PCHIP [4]
Computation time (s)	2.35	7
l	6009.30	6020
ns	20	18
nn	21	–
$T_{Transfer}$	41.37	41.50

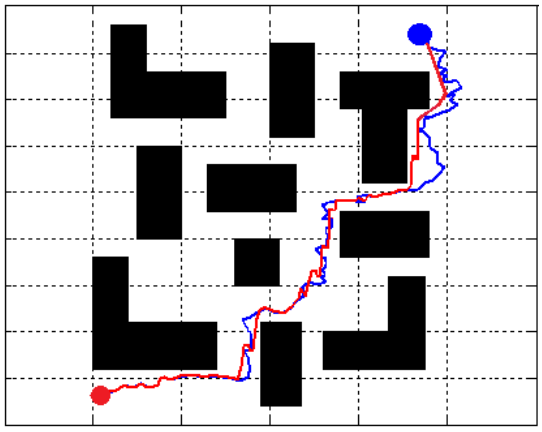
5.3.2. Comparison with *Conventional RRT*

For this comparison, the robot evolves in complex environment described in Fig. 8 [5]. The mobile robot has to move from $Source(x, y)$ = $(0mm, 0mm)$ toward$Target(x, y)$ = $(10000mm, 10000mm)$. Fig. 8(a) shows the path generated by *RRT-A*-BT*; Fig. 8(b) gives the result returned by *Conventional RRT* [5].

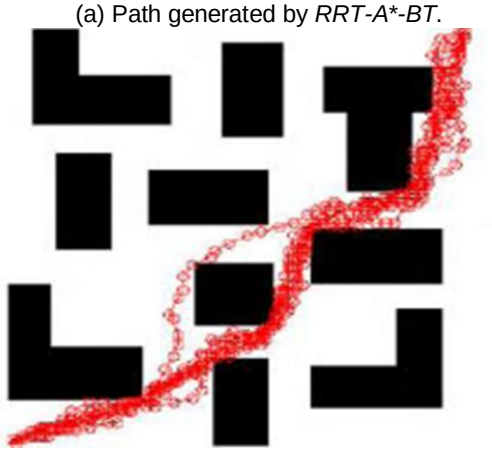
Table 3: Comparison between *Conventional RRT* [5] and the proposed *RRT-A*-BT*.

Parameters	RRT-A*-BT	RRT [5]
Computation time (s)	1.87	2.02
l	168.30	179.90
ns	20	–
nn	21	–

According to Table 3, it is obvious that *RRT-A*-BT* is better and more efficient than that developed in [5]. Additionally, to generate the path connecting $Source$to$Target$ positions, *RRT-A*-BT* is faster than *Standard RRT* algorithm with a minimum path length.



(a) Path generated by *RRT-A*-BT*.



(b) Path generated by *Standard RRT* [5].

Fig. 8. *RRT-A*-BT* versus *RRT* developed in [5].

5.4. Other comparisons

To show the effectiveness of our approach in terms of computation times, different workspaces with different obstacles are considered. Each case considers different numbers of cameras with different $Source - Target$ positions. Table 4 summarizes the average times of many runs of *RRT-A*-BT* and *GA-PCHIP* algorithms.

Table 4: Comparison between *RRT-A*-BT* and *GA-PCHIP* [4] for different workspaces, dimensions and obstacles.

Resolution	no	Computation time (s)	
		RRT-A*-BT	GA-PCHIP [4]
640*480 (01 Kinect camera)	05	1.25	5.60
	10	2.87	6.30
	15	3.99	7.50
640*960 (02 Kinect cameras)	10	2.01	7.42
	15	3.23	8.74
	20	4.62	9.06
960*1280 (03 Kinect cameras)	15	4.12	10.30
	20	4.24	10.56
	25	5.73	11.26

Table 4 shows that *RRT-A*-BT* is better and more efficient than *GA-PCHIP*. Moreover, its computation times in all considered environments are better than *GA-PCHIP*. Despite the fact that *BT* algorithm requires an important stack memory space to store *BT*nodes besides additional calculation time and resources to manage this structure [56], obtained computation times are very satisfactory. Finally, the computational results show the effectiveness of *RRT-A*-BT* and comparison with other approaches of the literature attest its efficiency and prove that it could even provide better results in complex workspaces.

6. CONCLUSIONS AND FUTURE WORKS

This article described an optimal collision-free path planning approach for mobile robots based on *RRT*, *A** and *BT* algorithms. *RRT-A*-BT* approach has been implemented to rapidly generate best paths connecting initial ($Source$) to final imposed ($Target$) configurations, ensuring that the robot will safely move between them. Additionally, several validation scenarios were performed and compared with other similar works (*RRT* and *GA-PCHIP*) while considering same environments and conditions. Results confirmed that the proposed *RRT-A*-BT* is better than *RRT* [5] and *GA-PCHIP* [4] in terms of (*i*) *Computation time*, (*ii*) *Path length*, (*iii*) *Segments number* and (*iv*) *Transfer time*. Future extensions can be listed as follows. First, *RRT-A*-BT* will be validated in more complex environments. Finally, this approach will be extended to deal with online path planning problems with larger workspaces.

REFERENCES

[1] S. G. Tzafestas, Introduction to mobile robot control. Elsevier, 2013.

[2] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," IEEE Transactions on Industrial Electronics, 2018.

[3] A. Hentout, A. Maoudj, D. Guir, S. Saighi, M. A. Harkat, M. Z. Hammouche, and A. Bakdi, "Collision-free path planning for indoor mobile robots based on rapidly-exploring random trees and piecewise cubic hermite interpolating polynomial," International Journal of Imaging and Robotics, vol. 19, no. 03, 2019.

[4] A. Bakdi, A. Hentout, H. Boutamai, A. Maoudj, O. Hachour, and B. Bouzouia, "Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control," Robotics and Autonomous Systems, vol. 89, pp. 95–109, 2017.

[5] R. Seif and M. A. Oskoei, "Mobile robot path planning by RRT* in dynamic environments," International Journal of Intelligent Systems and Applications, vol. 7, no. 5, p. 24, 2015.

[6] I. K. Ibraheem and F. H. Ajeil, "Path planning of an autonomous mobile robot in a dynamic environment using modified bat swarm optimization," arXiv preprint arXiv:1807.05352, 2018.

[7] J.-w. Choi, R. Curry, and G. Elkaim, "Real-time obstacle-avoiding path planning for mobile robots," in AIAA Guidance, Navigation, and Control Conference, 2010, p. 8411.

[8] N. Habib, D. Purwanto, and A. Soeprijanto, "Mobile robot motion planning by point to point based on modified ant colony optimization and voronoi diagram," in Intelligent Technology and Its Applications (ISITIA), 2016 International Seminar on. IEEE, 2016, pp. 613–618.

[9] C. Li, B. Bodkin, and J. Lancaster, "Programming Khepera II robot for autonomous navigation and exploration using the hybrid architecture," in Proceedings of the 47th Annual Southeast Regional Conference. ACM, 2009, p. 31.

[10] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. IEEE, 1991, pp. 1398–1404.

[11] A. Pandey, R. K. Sonkar, K. K. Pandey, and D. Parhi, "Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller," in Intelligent Systems and Control (ISCO), 2014 IEEE 8th International Conference on. IEEE, 2014, pp. 39–41.

[12] J. Yu and V. Kromov, "A rapid path planning algorithm of neural network," Robot, vol. 23, no. 3, pp. 201–205, 2001.

[13] P. Das, S. Pradhan, S. Patro, and B. Balabantaray, "Artificial immune system based path planning of mobile robot," in Soft Computing Techniques in Vision Science. Springer, 2012, pp. 195–207.

[14] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in Music-inspired harmony search algorithm. Springer, 2009, pp. 1–14.

[15] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," Robotics and Autonomous Systems, vol. 64, pp. 137–141, 2015.

[16] R. R. Yacoub, R. T. Bambang, A. Harsoyo, and J. Sarwono, "DSP implementation of

combined fir-functional link neural network for active noise control," International Journal of Artificial Intelligence, vol. 12, no. 1, pp. 36–47, 2014.
[17] M. K. Rath and B. Deepak, "PSO based system architecture for path planning of mobile robot in dynamic environment," in Communication Technologies (GCCT), 2015 Global Conference on. IEEE, 2015, pp. 797–801.
[18] C. Purcaru, R.-E. Precup, D. Iercan, L.-O. Fedorovici, and R.-C. David, "Hybrid PSO-GSA robot path planning algorithm in static environments with danger zones," in System theory, control and computing (ICSTCC), 2013 17th international conference. IEEE, 2013, pp. 434–439.
[19] B. B. K. Ayawli, R. Chellali, A. Y. Appiah, and F. Kyeremeh, "An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning," Journal of Advanced Transportation, vol. 2018, 2018.
[20] I. K. Ibraheem and F. H. Ajeil, "Multi-objective path planning of an autonomous mobile robot in static and dynamic environments using a hybrid PSO-MFB optimisation algorithm," arXiv preprint arXiv:1805.00224, 2018.
[21] A. Pandey and D. R. Parhi, "Autonomous mobile robot navigation in cluttered environment using hybrid Takagi-Sugenofuzzy model and simulated annealing algorithm controller," World Journal of Engineering, vol. 13, no. 5, pp. 431–440, 2016.
[22] A. Almayyahi, W. Wang, A. A. Hussein, and P. Birch, "Motion control design for unmanned ground vehicle in dynamic environment using intelligent controller," International Journal of Intelligent Computing and Cybernetics, vol. 10, no. 4, pp. 530–548, 2017.
[23] M. Saraswathi, G. B. Murali, and B. Deepak, "Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm," Procedia computer science, vol. 133, pp. 510–517, 2018.
[24] P. K. Mohanty and D. R. Parhi, "Cuckoo search algorithm for the mobile robot navigation," in International Conference on Swarm, Evolutionary, and Memetic Computing. Springer, 2013, pp. 527–536.
[25] S. Ghosh, P. K. Panigrahi, and D. R. Parhi, "Analysis of FPA and BA meta-heuristic controllers for optimal path planning of mobile robot in cluttered environment," IET Science, Measurement & Technology, vol. 11, no. 7, pp. 817–828, 2017.
[26] P. Das, H. Behera, P. Jena, and B. Panigrahi, "Multi-robot path planning in a dynamic environment using improved gravitational search algorithm," Journal of Electrical Systems and Information Technology, vol. 3, no. 2, pp. 295–313, 2016.
[27] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT* based approaches: a survey and future directions," International Journal of Advanced Computer Science and Applications, vol. 7, no. 11, pp. 97–107, 2016.
[28] O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree based memory efficient motion planning," in Mechatronics and Automation (ICMA), 2013 IEEE International Conference on. IEEE, 2013, pp. 354–359.
[29] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on. IEEE, 1994, pp. 2138–2145.
[30] L. E. Kavraki, P. Svestka, J.-C. 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.
[31] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.
[32] A. Hentout, H. Lehtihet, T. Chettibi, and B. Bouzouia, "Roadmap-based collision-free trajectory planning for manipulator robots," Journal of Modelling & Simulation of Systems, vol. 1, no. 1, pp. 40–49, 2010.
[33] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," Robotics and Autonomous Systems, vol. 68, pp. 1–11, 2015.
[34] I. Garcia and J. P. How, "Improving the efficiency of rapidly-exploring random trees using a potential function planner," in Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on. IEEE, 2005, pp. 7965–7970.
[35] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, vol. 2. IEEE, 2000, pp. 995–1001.
[36] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on. IEEE, 2009, pp. 2222–2229.
[37] 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 (IJCSNS), vol. 16, no. 10, p. 20, 2016.
[38] A. Hentout, B. Bouzouia, R. Toumi, and Z. Toukal, "Agent-based coordinated control of mobile manipulators," in The International Conference on Systems and Processing Information (ICSIP'09). University of Guelma, Algeria, 2009.
[39] A. Maoudj, A. Hentout, B. Bouzouia, and R. Toumi, "On-line fault-tolerant fuzzy-based path planning and obstacles avoidance approach for manipulator robots," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 26, no. 5, pp. 809–839, 2018.
[40] L. Knispel and R. Matousek, "A performance comparison of rapidly-exploring random tree and Dijkstrasalgorithm for holonomic robot path planning," Institute of Automation and Computer Science, Faculty of Mechanical Engineerig, Brno University of Technology, 2013.
[41] L. Kang, C.-X. Zhao, and J.-H. Guo, "Improved path planning based on rapidly-exploring random tree for mobile robot in unknown environment [j]," Pattern

Recognition and Artificial Intelligence, vol. 3, 2009.
[42] R. Kala, "Rapidly exploring random graphs: motion planning of multiple mobile robots," Advanced Robotics, vol. 27, no. 14, pp. 1113–1122, 2013.
[43] J. Nieto, E. Slawinski, V. Mut, and B. Wagner, "Online path planning based on rapidly-exploring random trees," in Industrial Technology (ICIT), 2010 IEEE International Conference on. IEEE, 2010, pp. 1451–1456.
[44] A. A. Neto, D. G. Macharet, and M. F. Campos, "Multi-agent rapidly-exploring pseudo-random tree," Journal of Intelligent & Robotic Systems, vol. 89, no. 1-2, pp. 69–85, 2018.
[45] M. Kothari, I. Postlethwaite, and D.-W. Gu, "A suboptimal path planning algorithm using rapidly-exploring random trees." International Journal of Aerospace Innovations, vol. 2, 2010.
[46] F. Duchon, A. Babinec, M. Kajan, P. Beno, M. Florek, T. Fico, and L. Jurisica, "Path planning with modified a star algorithm for a mobile robot," Procedia Engineering, vol. 96, pp. 59–69, 2014.
[47] F. Duchon, D. Hunady, M. Dekan, and A. Babinec, "Optimal navigation for mobile robot in known environment," in Applied Mechanics and Materials, vol. 282. Trans Tech Publ, 2013, pp. 33–38.
[48] T.-K. Lee, S.-H. Baek, Y.-H. Choi, and S.-Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," Robotics and Autonomous Systems, vol. 59, no. 10, pp. 801–812, 2011.
[49] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "Ba*: an online complete coverage algorithm for cleaning robots," Applied intelligence, vol. 39, no. 2, pp. 217–235, 2013.
[50] Z. Aljarboua, "Geometric path planning for general robot manipulators," in Proceedings of the world congress on engineering and computer science, vol. 2. Citeseer, 2009, pp. 20–22.
[51] M. Lepetic, G. Klancar, I. Skrjanc, D. Matko, and B. Potoˇ cnik, "Time optimal path planning considering acceleration limits," Robotics and Autonomous Systems, vol. 45, no. 3-4, pp. 199–210, 2003.
[52] E. Papadopoulos, I. Papadimitriou, and I. Poulakakis, "Polynomial-based obstacle avoidance techniques for non-holonomic mobile manipulator systems," Robotics and Autonomous Systems, vol. 51, no. 4, pp. 229–247, 2005.
[53] I. Skrjanc and G. Klancar, "Optimal cooperative collision avoidance between multiple robots based on bernstein–bezier curves," Robotics and Autonomous systems, vol. 58, no. 1, pp. 1–9, 2010.
[54] H. Belaidi, A. Hentout, B. Bouzouia, H.Bentarzi and A. Belaidi, "NURBs trajectory generation and following by an autonomous mobile robot navigating in 3D environment," in The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent. IEEE, 2014, pp. 168-173.
[55] A. Hentout, M. R. Benbouali, I. Akli, B. Bouzouia, and L. Melkou, "A telerobotic human/robot interface for mobile manipulators: A study of human operator performance," in Control, Decision and Information Technologies (CoDIT), 2013 International Conference on. IEEE, 2013, pp. 641–646.
[56] Y.-H. Choi, T.-K. Lee, S.-H. Baek, and S.-Y. Oh, "Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform," in Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. IEEE, 2009, pp. 5788–5793.