This article can be cited as A. Hentout, A. Maoudj, D. Guir, S. Saighi, M. A. Harkat, M. Z. Hammouche and A. Bakdiy, 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. 3, pp. 74-97, 2019.

Collision-free Path Planning for Indoor Mobile Robots Based on Rapidly-exploring Random Trees and Piecewise Cubic Hermite Interpolating Polynomial

Abdelfetah Hentout[1], Abderraouf Maoudj[2], Dallel Guir[3], Souhila Saighi[4], Mohamed Aures Harkat[5], Mohamed Zakaria Hammouche[6] and Azzeddine Bakdi[7] *†

[1,2,3,4,5,6]Centre de Développement des Technologies Avancées (CDTA)
Division Productique et Robotique (DPR)
BP 17, Baba Hassen, Algiers 16303, Algeria.
ahentout@cdta.dz, amaoudj@cdta.dz

[3,4,7]University M'hamed Bougara de Boumerdès (UMBB)
Institute of Electrical and Electronic Engineering (IGEE)
Boulevard de l'Indépendance, Boumerdès, 35000, Algeria.
dallelgrl-93@hotmail.com, bkdaznsun@gmail.com

[5,6]University of Sciences and Technology Houari Boumediene (USTHB)
Faculty of Electronics and Computer Science
BP 32 El Alia, Bab Ezzouar, Algiers 16111, Algeria.
harkatmohamedaures@gmail.com, mhaalansari2016@gmail.com

ABSTRACT

This paper deals with the off-line path planning problem for wheeled indoor mobile robots. In the proposed approach, the robot exploits depth information acquired by a two-Kinect cameras vision system to perfectly model its surroundings environment. Then, the *Rapidly-exploring Random Tree* algorithm is used to generate a collision-free path linking the initial configuration ($Source$) to the final configuration ($Target$). This feasible path consists of a set of n nodes (i.e., $n-1$ edges) in form of (x, y) vectors. Next, an algorithm is proposed to reduce the number of nodes and edges of the generated path. Finally, the found path is smoothed using *Piecewise Cubic Hermite Interpolating Polynomial* technique. The proposed *RRT-PCHIP* solution is validated on the differential mobile robot, *RobuTER*, evolving in an indoor environment cluttered with static obstacles. Obtained results are compared with another similar work using a *Genetic Algorithm* in terms of (*i*) *generation time*, (*ii*) *path length*, (*iii*) *number of segments constituting the path* and (*iv*) *transfer time*.

Keywords: Mobile robots, Off-line path planning, *Rapidly-exploring Random Tree*, *Piecewise cubic Hermite Interpolating Polynomial*, Two-Kinect system, *Robot Operating System*.

*[3,4]Both authors contributed equally to this work.
†[5,6]Both authors contributed equally to this work.

2000 Mathematics Subject Classification: 68T40, 68U10.
ACM Computing Classification System: Computing methodologies→Motion path planning; Vision for robotics;

1 Introduction

Path planning problem is one of the most important issues in autonomous mobile robotics field in either 2D or 3D workspaces (Sureshkumar, Mahalingam and Rama, 2017). It mainly tries to identify a series of configurations that will safely move the robot (i.e., an obstacle-free path) from an initial configuration ($Source$) to a final configuration ($Target$) (Fahimi, 2008) (Tzafestas, 2013).

Many authors have proposed solutions for path planning problems (Das, Behera, Jena and Panigrahi, 2016) using different classical techniques (Li, Bodkin and Lancaster, 2009), neural network (Yu, Kromov et al., 2001), artificial immune system (Das, Pradhan, Patro and Balabantaray, 2012), heuristic optimization algorithms (Yang, 2009), etc. Other authors proposed many evolutionary and differential evolution algorithms to deal with such a problem. Hossain and Ferdous (Hossain and Ferdous, 2015) explored the application of *Bacterial Foraging Optimization* (*BFO*) for mobile robot path planning, through simulation application. The obstacles are wrapped by circles, and the robot by a square (i.e., low accuracy); the generation of a safe path requires dozens of seconds and depends on the number of obstacles, better than basic *BFO* and *Particle Swarm Optimization* (*PSO*) algorithms (Dhariwal, Sukhatme and Requicha, 2004). This latter algorithm has been adopted by a considerable number of researchers in mobile robot path planning problems (Yacoub, Bambang, Harsoyo and Sarwono, 2014) (Rath and Deepak, 2015) (Ma, Wang, Xie and Guo, 2014). It has also been combined with algorithms such as *Gravitational Search* (*GS*) (Purcaru, Precup, Iercan, Fedorovici and David, 2013), *Artifcial Potential Field* (*APF*) (Ayawli, Chellali, Appiah and Kyeremeh, 2018) and *Modified Frequency Bat* (*MFB*) (Ibraheem and Ajeil, 2018). Cabreira and colleagues (Cabreira, Dimuro and de Aguiar, 2012) (Cabreira, de Aguiar and Dimuro, 2013) used a *Genetic Algorithm* (*GA*) with a very limited resolution due to cell decomposition of their simulated environment. Moreover, neither exact robot geometry and location nor obstacles location and shapes were taken into account while dealing with such a problem. The *Fuzzy Logic* (*FL*) control system exhibiting a good dynamic performance (Haidegger, Kovács, Precup, Benyó, Benyó and Preitl, 2012) is widely used by many approaches in the literature. Bakdi et al. (Bakdi, Hentout, Boutamai, Maoudj, Hachour and Bouzouia, 2017) developed an off-line collision-free path planning and execution approach based on *GA* and *FL* control using a two-Kinect system. Similarly, the path planning problem in cluttered environments was investigated using hybrid Takagi-Sugeno *FL* model and the *Simulated Annealing* (*SA*) algorithm controller (Pandey and Parhi, 2016) (Vrkalovic, Teban and Borlea, 2017). The experiment was performed in simple environments; it showed an efficient navigation (Almayyahi, Wang, Hussein and Birch, 2017).

The aforementioned techniques are described to be simple and effective in implementing with varied path planning problems with good results (Ma, Zamirian, Yang, Xu and Zhang, 2013). However, the high time complexity in large problem spaces and trapping into local minima under complex environments are the main drawbacks of these classical techniques and many

meta-heuristic algorithms (Su, Wang and Hu, 2015). These drawbacks cause the classical techniques to be inefficient in various problem spaces (Das et al., 2016).

In order to overcome these disadvantages and improve the efficiency of classical methods, *Sampling-Based Planners* (*SBP*) have been developed these last decades. These algorithms have proven to be computationally efficient solutions (Canny, 1988) and have become de facto standard for high-dimension motion planning problems (Adiyatov and Varol, 2013). The most well-known *SBP* algorithms include *Probabilistic Roadmap Methods* (*PRMs*) (Kavraki and Latombe, 1994) (Kavraki, Svestka, Latombe and Overmars, 1996) and *Rapidly-exploring Random Trees* (*RRTs*) (LaValle, 1998). *PRMs* were successfully used with high-dimension workspaces by several researches (Hentout, Lehtihet, Chettibi and Bouzouia, 2010); however, they tend to be inefficient when obstacle geometry is not known beforehand (Karaman and Frazzoli, 2011) (Qureshi and Ayaz, 2015).

Consequently, to generate efficient solutions for motion planning, *RRT* algorithms have been extensively explored. In addition, several variants enhancing the original *RRT* algorithm have been proposed (Lindemann and LaValle, 2004) (Garcia and How, 2005). One of these variants, *RRT-connect* algorithm proposed in (Kuffner and LaValle, 2000), checks if traveling straight in the line of expanded point reaches $Target$; in such a case, the algorithm terminates. Another variant is *Bidirectional RRT* (LaValle and Kuffner Jr, 2001) which grows two trees, one rooted at the initial configuration $Source$ and the other rooted at $Target$; the process ends when the two trees meet. The last variant, *RRT**, combines the standard *RRT* with *Rapidly-exploring Random Graph* (*RRG*) algorithm (Karaman and Frazzoli, 2009) to provide asymptotically-optimal solutions linking $Source$ to $Target$ (Karaman and Frazzoli, 2010) (Jordan and Perez, 2013). *RRT** improved the path quality by introducing major features of the tree rewiring and best neighbor search (Noreen, Khan and Habib, 2016).

RRTs are proposed as both data structure and sampling algorithm designed for efficient and fast search in non-convex high-dimension spaces in path planning problems with state constraints (involving obstacles, etc.) (LaValle and Kuffner Jr, 2001) (LaValle, 2006) (Knispel and Matousek, 2013). *RRTs* are progressively built as a tree using random sampling toward unexplored regions of the workspace. The search tree starts from $Source$, as the root, and expands to find a path toward $Target$ (Kuffner and LaValle, 2000) (LaValle and Kuffner Jr, 2001). At every iteration, the algorithm generates a random node in the *configuration space* ($C - Space$), searches for the nearest node in the tree and extends this node to the random sample by a predefined step size (Kang, Zhao and Guo, 2009) (Vonásek, Faigl, Krajník and Přeučil, 2009) (Kuwata, Fiore, Teo, Frazzoli and How, 2008) (Pepy and Lambert, 2006) (Burns and Brock, 2007) (Kagami, Kuffner, Nishiwaki, Okada, Inaba and Inoue, 2003). The algorithm stops when (*i*) the exploration results in reaching $Target$ (best case) (Kala, 2013) or (*ii*) one of the leaves of the tree reaches a $Target$ region (in most cases).

The main advantages of *RRTs* are simplicity (since their implementation is straightforward and only needs simple computations), fast convergence and expansion toward unexplored regions of the $C - Space$, and probabilistically completeness (Neto, Macharet and Campos, 2018). Furthermore, *RRTs* are computationally efficient since they calculate solution paths in a fast way and are suitable for path planning in high-dimension spaces (Nieto, Slawinski, Mut and

Wagner, 2010). However, *RRTs* are suboptimal (Kothari, Postlethwaite and Gu, 2010) which may be global or local. *Global optimality* indicates the strategy to avoid obstacles (whether to go from left/right or in-between obstacles); whereas *local optimality* indicates distances to maintain from obstacles (Kala, 2013).

This work deals with the development of a strategy combining *Rapidly-exploring Random Trees (RRT)* with *Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)*, *RRT-PCHIP*, to tackling the off-line collision-free path planning problem for mobile robots which evolve in indoor environments cluttered with static obstacles. The main contributions of this work can be outlined as follows. First, the proposed approach deals with global path planning with exact highly-constrained environment modeling. Second, the robot makes use of two-Kinect vision system with high accuracy to model its surrounding constrained workspace through image processing. It also ensures exact representation of the robot and obstacles considering their shapes and locations. Third, the *RRT* algorithm is used for global planning and to generate a feasible collision-free path represented as a set of nodes and connected segments linking the $Source$ to the imposed final $Target$. Fourth, an algorithm has been proposed to reduce the number of nodes and edges of the found path. Fifth, the use of *PCHIP* (Hyman, 1983) for path smoothing increases the average linear velocity of the robot for faster execution. Sixth, the left (V_L) and right (V_R) velocities are calculated for a differential mobile robot, *RobuTER*, to track this desired feasible path. Finally, all the above contributions have been done using *Robot Operating System* (*ROS*) framework (*ROS*, 2018).

The comparative study allows affirming that the proposed *RRT-PCHIP* approach is better than other methods in the literature, such as *GA-PCHIP* presented in (Bakdi et al., 2017), especially in terms of (*i*) generation time, (*ii*) number of segments of the feasible path and (*iii*) transfer time of the robot. Consequently, it can be easily extended to deal with on-line path planning problems for non-static environments, with larger workspaces for the robots.

The remainder of this paper is organized as follows. Section 2 gives a short description of the path planning problem for non-holonomic mobile robots in environments with obstacles. Section 3 summarizes the overall approach and describes the experimental robotic system. Section 4 describes the proposed collision-free path planning approach based on *Rapidly-exploring Random Trees* and path smoothing using *Piecewise Cubic Hermite Interpolating Polynomial*. Section 5 presents, discusses and compares the main obtained experimental results. Section 6 concludes the paper and presents some future works.

2 Path planning problem statement

This work aims to propose an efficient path planning algorithm to guarantee safe navigation of the mobile robot from a given initial position $Source(x, y, \theta)$ to an imposed final position $Target(x, y, \theta)$ in known indoor environments. This algorithm takes into account the mechanical constraints of the mobile robot (kinematic limits) and the presence of obstacles in its workspace (collision avoidance).

The path planning algorithms for non-holonomic mobile robots are based on robot kinematic model (Ma, Zheng, Perruquetti and Qiu, 2015) given as follows (Hentout, Bouzouia, Toumi and

Toukal, 2009):

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{2.1}$$

where:

- $q = (x, y, \theta)$ is the robot state,
- $Source(x, y, \theta)$ and $Target(x, y, \theta)$ represent the initial and final configurations of the mobile robot, respectively,
- v is the linear velocity of the robot,
- ω is the angular velocity of the robot,
- θ is the orientation of the mobile robot body with respect to $x-$axis.

Generally, the evolution environments of the mobile robots may be complex and normally obstacles cannot be described as circles, as assumed by Defoort and colleagues (Defoort, Palos, Kokosy, Floquet and Perruquetti, 2009). Fig. 1 shows a mobile robot in an encumbered workspace with many obstacles represented in an accurate way (exact representation of the obstacles). The different constraints that must be satisfied during the entire transfer of the mobile robot ($0 \leq t \leq T_{Transfer}$) are summarized as follows by equations 2.2 ... 2.6 (adapted from (Maoudj, Hentout, Bouzouia and Toumi, 2018)):

$$q(t = 0) = Source(x, y, \theta) \tag{2.2}$$

$$q(t = T_{Transfer}) = Target(x, y, \theta) \tag{2.3}$$

$$v(t) \in [v_{min}, v_{max}], t \in [0, T_{Transfer}] \tag{2.4}$$

$$\omega(t) \in [\omega_{min}, \omega_{max}], t \in [0, T_{Transfer}] \tag{2.5}$$

$$Collision(Robot, Obstacle_i) = \emptyset, i = 1 \ldots n \tag{2.6}$$

where:

- $T_{Transfer}$ represents the final transfer time,
- v_{min} and v_{max} are the minimum and maximum linear velocities of the mobile robot,
- ω_{min} and ω_{max} are the minimum and maximum angular velocities of the mobile robot,
- $Collision$ is a *Boolean* function that indicates whether or not the mobile robot is in collision with an obstacle $Obstacle_i$ ($i \in \{1 \ldots n\}$),
- n is the number of obstacles present in the environment. We assume that the obstacles do not overlap neither with $Source$ nor $Target$.

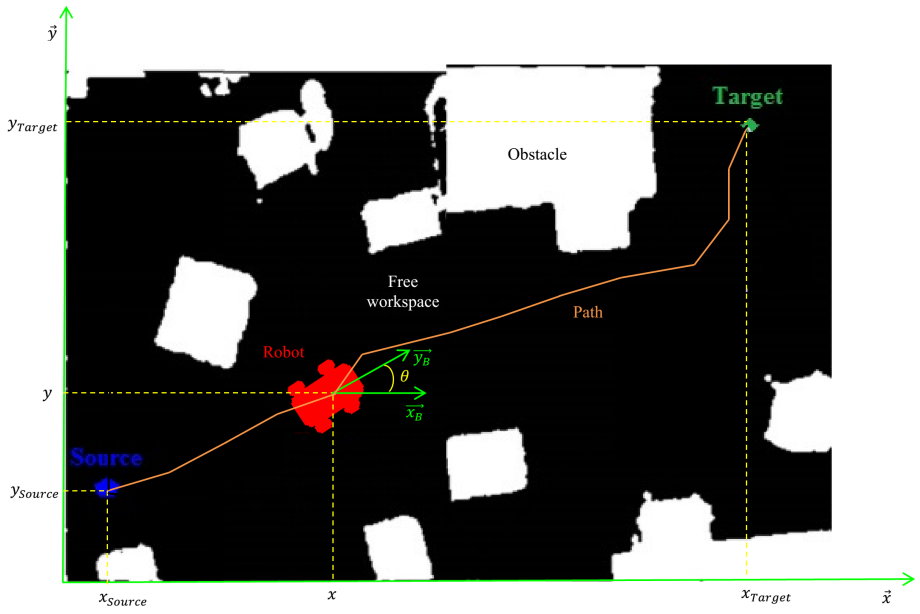


Figure 1: Path planning for wheeled mobile robots in presence of obstacles.

3 Global overview of the proposed approach

The main objective of this work is to plan a collision-free smoothed path from $Source(x, y, \theta)$ to $Target(x, y, \theta)$ using the *RRT* algorithm. The mobile robot evolves inside a cluttered workspace modeled by using a two-Kinect cameras vision system.
Fig. 2 summarizes the overall diagram of the proposed approach for off-line collision-free path planning. It consists of six successive phases:

1. *Phase (1)*: Visual perception of robot environment,
2. *Phase (2)*: Processing of acquired images and modeling of the robot workspace,
3. *Phase (3)*: Planning a collision-free path using *RRT* approach and reduction of the number of edges and nodes of this path,
4. *Phase (4)*: Smoothing this path using the *PCHIP* technique,
5. *Phase (5)*: Sensor-fusion localization of the mobile robot using *Kalman Filter*, and finally
6. *Phase (6)*: Path tracking and execution by exploiting the *Adaptive FL Controller* developed in (Bakdi et al., 2017).

The two last phases are carried out in parallel. It must be noted that this paper only deals with phases (1), (2), (3) and (4); the other phases ((5) and (6)) have already been detailed in (Bakdi et al., 2017).

4 Path planning and execution approach

As described above, the proposed approach consists of several phases. Only phases (1), (2), (3) and (4) are detailed in the following sub-sections.

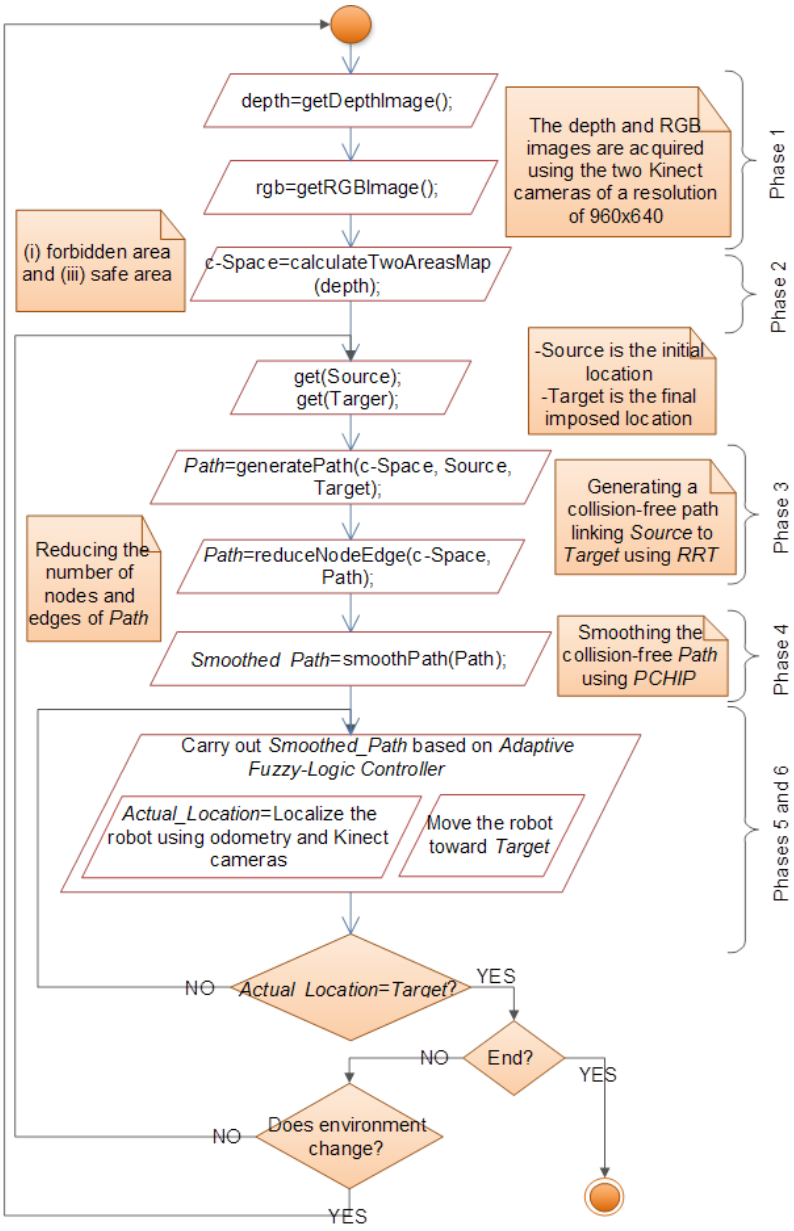


Figure 2: Overall diagram of the proposed approach for collision-free path planning and execution.

4.1 Phases (1) and (2): Perception and modeling of the environment using two-Kinect vision system

Nowadays, there are many sensors that allow robots to perceive their evolution workspaces such as vision, laser and ultrasound sensors (Soriano, Bernabeu, Valera and Vallés, 2014). In this work, the visual perception of the robot environment is done using two *Microsoft Kinect* cameras system *Version 1*.

The Kinect has two sensors, a color sensor and a depth sensor. The image stream contains color data in RGB format with a resolution of 640×480 and 30 *frames per second* (*fps*). The depth stream returns depth information in pixels (same resolution and rate) with a valid range of $[500mm, 4000mm]$. For more information, refer to (Sgorbissa and Verda, 2013).

As appeared in Fig. 3, two Kinect cameras are fixed on the roof of the workroom; each Kinect is placed at a height of $height = 3500\ mm$. They are set in such a way with the same axis to visualize both the environment (ground, obstacles, etc.) and the robot. The Kinect cameras are connected to the off-board PC to acquire and process images. The resolution of the pictures delivered by the two-Kinect vision system is 640×960 pixels and covers a range of $4058\ mm \times 6087\ mm$, approximately. The horizontal accuracy is $1\ pixel \approx 6.34\ mm$; the vertical accuracy $(depth) = 1\ mm$. The following snapshots show the results of these two phases which consist

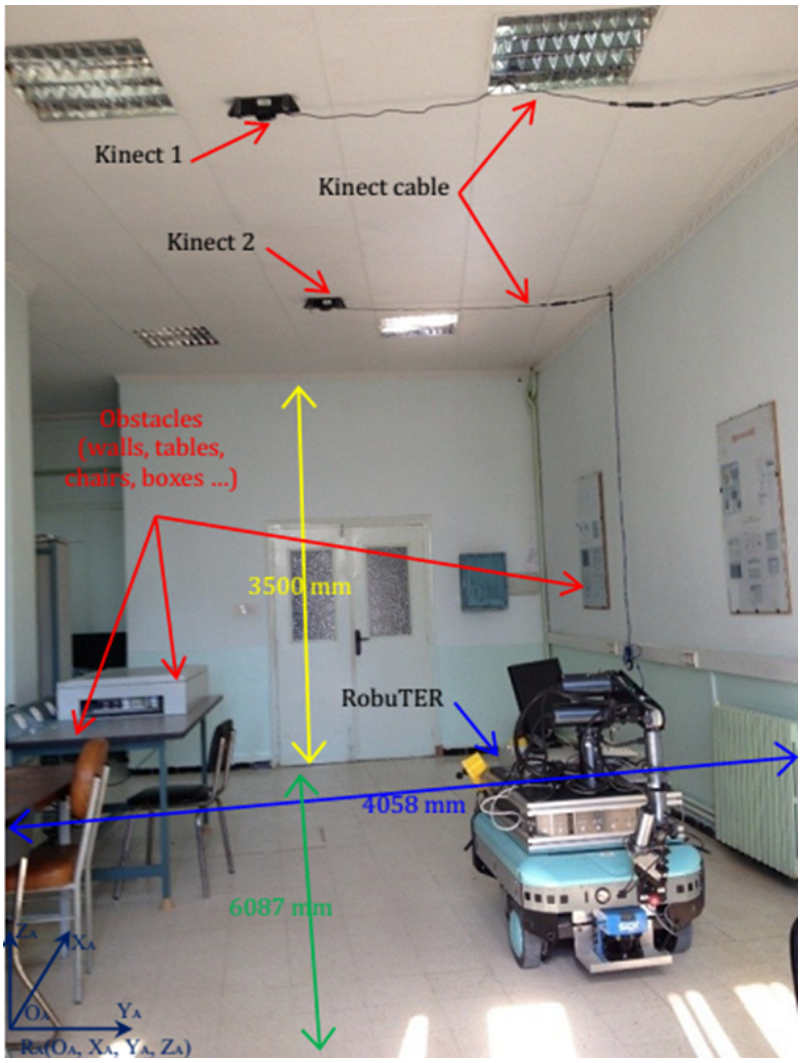
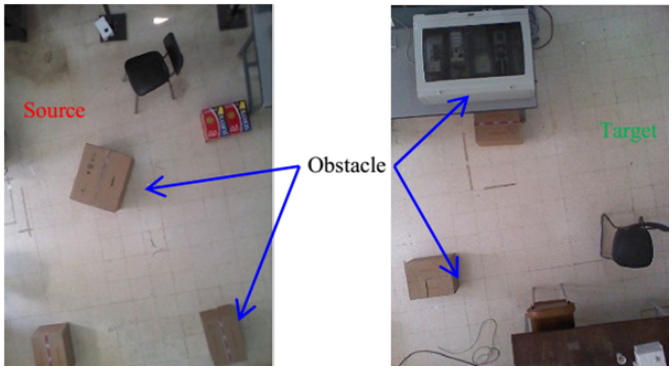


Figure 3: Experimental robotic test bed.

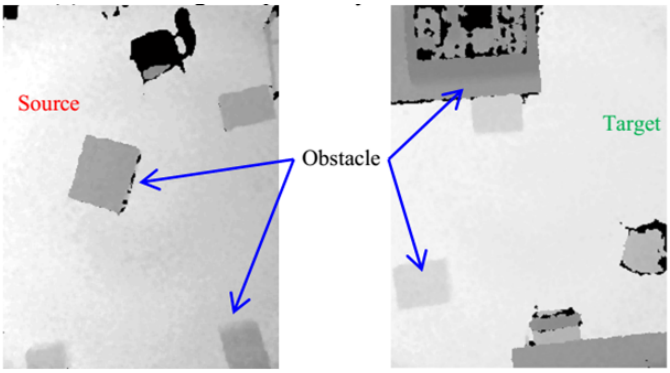
of an almost continuous description of the environment. Fig. 4(a) represents the acquired RGB images by the first and second Kinect cameras, separately. Fig. 4(b) shows depth images for the considered environment of the robot given by Kinect 1 and Kinect 2, respectively. Finally, Fig. 4(c) shows the environment modeling representing two areas:

- *Forbidden area*: obstacles and near obstacles, and
- *Safe area* or *Free* $C - Space$: far away from the obstacles.

For more details on the visual perception of the robot indoor environment, images processing



(a) RGB images captured by Kinect 1 and Kinect 2.



(b) Depth images captured by Kinect 1 and Kinect 2.

(c) Obtained safe map (Kinect 1 + Kinect 2).

Figure 4: Visual perception of the environment and obtained processed workspace.

and environment modeling, please refer to (Bakdi et al., 2017).

4.2 Phase (3): Collision-free path planning and reduction of the number of nodes and edges

This phase generates first a collision-free path using the *RRT* algorithm as detailed in subsection 4.2.1. The found path consists of a succession of nodes connected by a set of segments. After that, the number of nodes and edges of this path is reduced since their number is impor-

tant. The proposed reduction algorithm is described in subsection 4.2.2.

4.2.1 Collision-free path planning using *RRT*

At the beginning of the algorithm, the random tree G is empty; $Source$ and $Target$ positions are thus introduced and the different configuration parameters $(MAX\_ITER, STEP, DIST)$ are set up:

- MAX_ITER: it represents the maximum of iterations (in this work, $MAX\_ITER = 10000$),
- $STEP$: it defines the maximum length of the edge connecting two successive nodes (in this work, $STEP = 8 \dots 60$ pixels),
- $DIST$: it is the radius of the search region near $Target$ (in this work, $DIST = 20$).

The objective is to randomly create the tree G for connecting these two locations. The condition $region(New, Target, DIST)$ tests if the new generated node New has reached a region centered in $Target$ (i.e., the two nodes are separated by a distance less than or equal to $DIST$) rather than evaluating if it exactly coincides with $Target$, which would take significantly much more iterations to reach (Nieto et al., 2010). While the above condition is not satisfied, the whole following process is carried out. First, a $Rand$ node is randomly generated in the $C - Space$ of the robot. Next, the nearest neighbor of $Rand$, already in the tree G, is searched by exploiting the *Euclidean distance* (calculated by equation 4.1 between $Node_1$ and $Node_2$). After that, the node $Near$ is extended in the direction of $Target$ by $STEP$ because the edges between nodes must not exceed the predefined $STEP$ size. This is an important parameter as it determines the edges length in the tree and, thus, the processing time. Smaller values imply more iterations of the algorithm to find $Target$; but also imply a better resolution. Bigger values can imply less iterations of the algorithm, but also much longer solution paths. The expansion is interrupted when the node New is in collision with obstacles or when it falls outside the boundary limits (in this case, $extendNode(Near, STEP)$ returns $NULL$). Indeed, during the construction of the tree, obstacles must be avoided by checking whether a given node lies inside an obstacle or not. Furthermore, each edge of the *RRT* (which corresponds to a path between two nodes) must entirely be in a free region. The boundary conditions are given by the physical limits of the $C - Space$ in which the robot evolves. Finally, the generated path can directly be found by following the predecessors of the last added node New toward $Source$ (Nieto et al., 2010).

$$EuclideanDistance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

where (x_1, y_1) and (x_2, y_2) represent the *Cartesian coordinates* of $Node_1$ and $Node_2$, respectively.

Algorithm 1, shown by Fig. 5, constructs an *RRT* in the $C - Space$ of the robot, and eventually searches a path between $Source$ and $Target$. $get(Parameter)$ gets configuration $Parameter$. $addNode(G, Node)$ adds a new $Node$ to the tree G. $region(New, Target, DIST)$ tests if two nodes New and $Target$ belong to the same region (i.e., the two nodes are separated by a

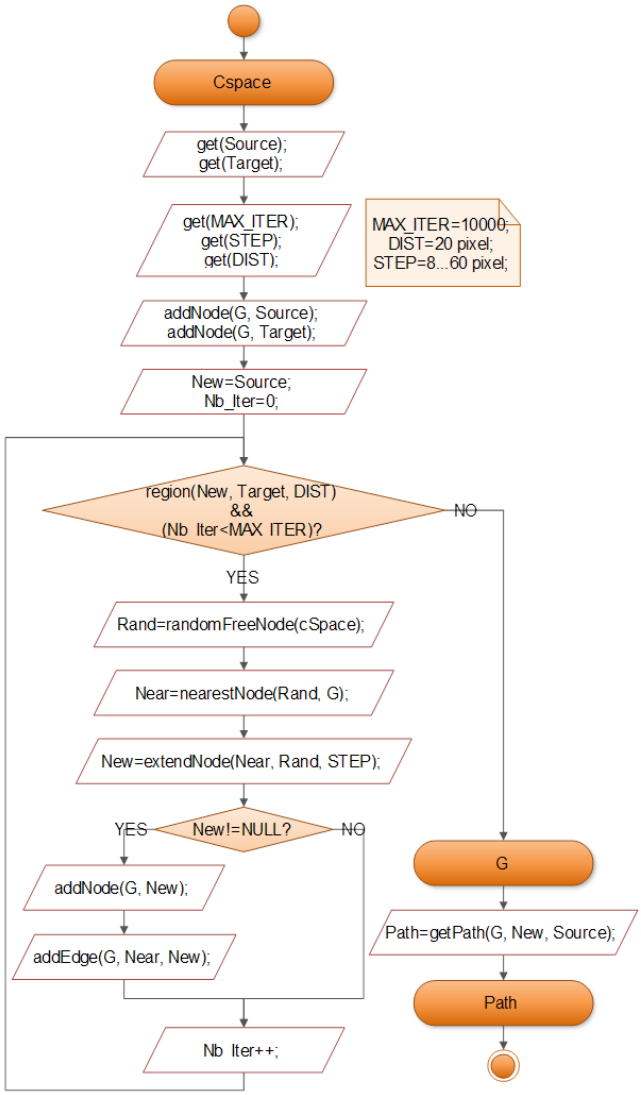


Figure 5: Algorithm 1. Description of the *RRT* algorithm for finding a collision-free path between $Source$ and $Target$.

distance less than or equal to $DIST$). $randomFreeNode(C-Space)$ generates a collision-free random configuration $Rand$ in the $C-Space$ of the robot. $addEdge(G, Near, New)$ adds a new edge from $Near$ to New in the tree G. $getPath(G, New, Source)$ allows to get the $Path$ in the tree G by following the predecessors of New toward $Source$.

The function $nearestNode(Rand, G)$ is described in Algorithm 2 shown by Fig. 6. It allows selecting the nearest node to $Node$ in the tree G in terms of *Euclidean distance*. $getSize(G)$ returns the size of the tree G. $getEuclideanDistance(Node_1, Node_2)$ calculates the *Euclidean distance* between two nodes $Node_1$ and $Node_2$. Algorithm 3, given by Fig. 7, describes the function $extendNode(Near, Rand, STEP)$. It returns configuration New by moving from $Near$ an incremental distance $STEP$ in the direction of $Rand$. $collision(New)$ tests if configuration New is in collision with obstacles or not. $outBoundary(New)$ verifies if New falls outside the boundary limits of the $C-Space$ of the mobile robot.

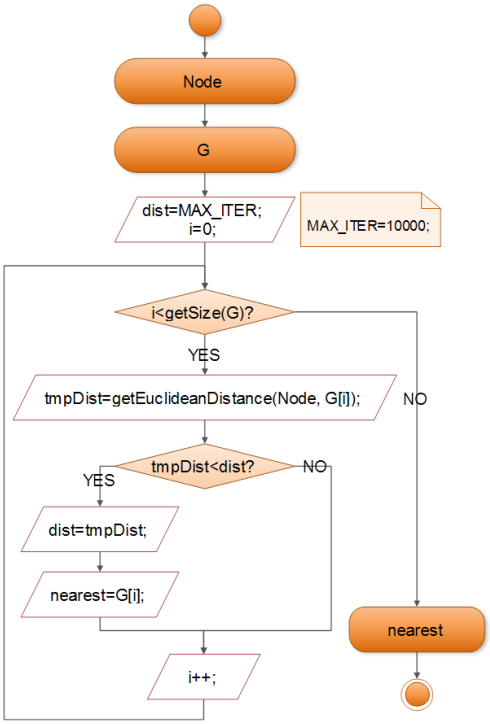


Figure 6: Algorithm 2. Searching for the nearest node of $Node$ in the tree G by using the *Euclidean distance*.

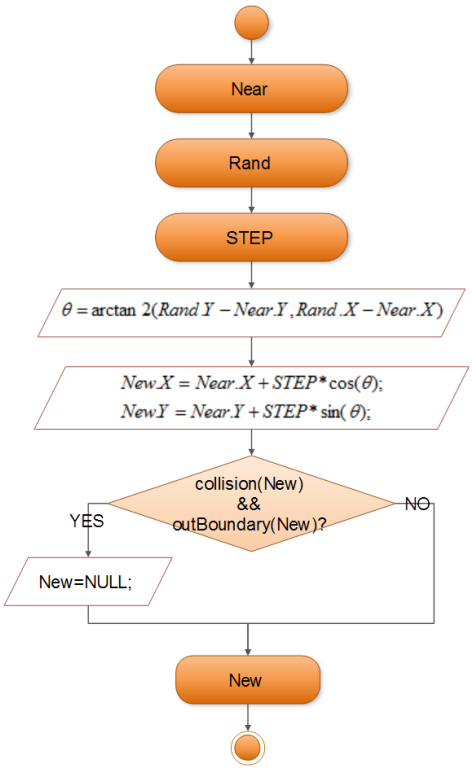


Figure 7: Algorithm 3. Selection of a node New by moving from $Near$ an incremental $STEP$ in direction of $Rand$.

4.2.2 Reducing the number of nodes and edges of the generated feasible path

The proposed reduction algorithm is based on testing the collision of a given edge of the generated path with the obstacles of the robot $C - Space$. Instead of verifying the collision between two successive nodes composing the same edge (which is useless), this verification is done between the first node of this edge ($Path[i]$) and the last node of the next edge ($Path[i+2]$). If no collision exists ($collision(Path[i], Path[i+2])$), a new segment is created between these two nodes and added to the new path $NewPath$ ($addEdge(NewPath, Path[i], Path[i+2])$); otherwise (i.e., a collision exists), the two edges are kept and added to $NewPath$. This process is reiterated until reaching $Target$ (i.e., last node of the path). Finally, all the previous procedure is repeated till no change is made on the new feasible path ($NewPath$).
Algorithm 4 (shown in Fig. 8) describes the principle of the proposed approach to reduce the total number of nodes and edges of the initially generated feasible path.

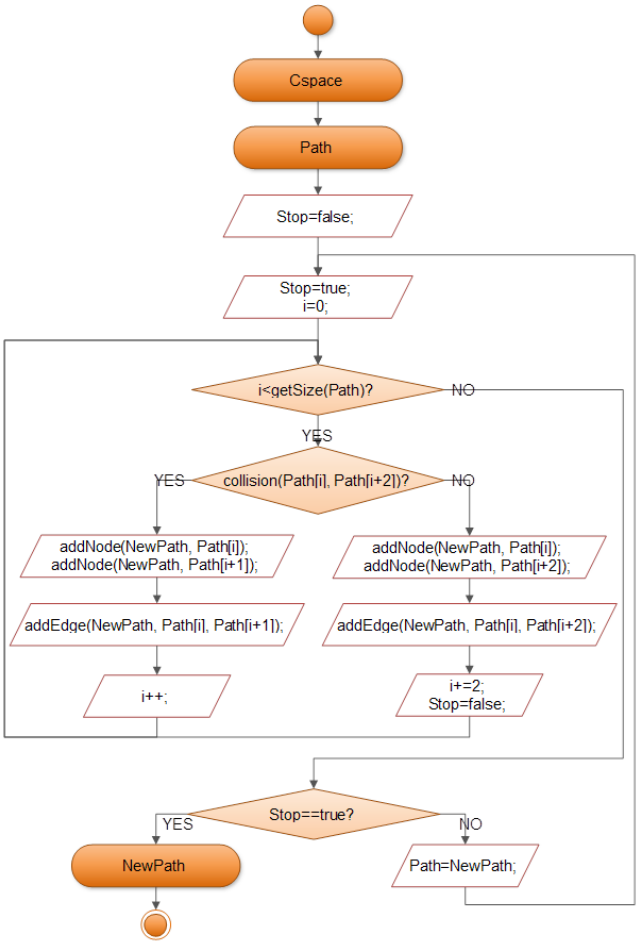


Figure 8: Algorithm 4. Reduction of the number of nodes and edges in the final collision-free path.

4.3 Phase (4): Path smoothing using *PCHIP*

The *RRT* algorithm stops when it finds a path connecting $Source$ to $Target$ without collision. This feasible path consists of a set of n nodes (i.e., $n-1$ edges or segments) in form of (x, y)

vectors as shown by Table 1. The $n-1$ connected segments need to be smoothed. For this purpose, the *PCHIP* approach (Hyman, 1983) is carried out on the resulted zigzag line path. In case of *PCHIP* interpolation, taking discrete data (i.e., the segments nodes), *PCHIP* generates the 3^{rd} order continuous polynomials (curves) instead of lines (1^{st} order) to smooth the path. The polynomials are found by solving the equations of nodes, continuity (derivatives) and minimal error (shape preserving), 3^{rd} order polynomial, four unknowns and four equations. Therefore, a new vector x_i of about 1000 points is generated from $Source$ to $Target$. $y_i = PCHIP(x, y, x_i)$ returns a vector y_i containing elements corresponding to the elements of x_i and determined by *piecewise cubic interpolation* within vectors x and y. Resulting y_i versus x_i give the smoothed path (Bakdi et al., 2017).

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

$Source$	2^{nd} **point**	3^{rd} **point**	**...**	$(n-1)^{th}$ **point**	$Target$
x_{Source}	x_2	x_3	...	x_{n-1}	x_{Target}
y_{Source}	y_2	y_3	...	y_{n-1}	y_{Target}

5 Experimental results

The path planning is performed by the proposed *RRT-PCHIP* approach and implemented using *ROS* framework. *ROS* is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms (*ROS*, 2018).

5.1 Description of the experimental robotic system

As Fig. 9 shows, the experimental robotic system is composed of two distant sites with wireless connection (Hentout, Benbouali, Akli, Bouzouia and Melkou, 2013):

- *Operator site* (*Local site*): it consists of an off-board host PC used to send orders to the robot using the wireless connection.
- *Remote site* (*Robot site*): it is a rectangular non-holonomic differential mobile manipulator robot, *RobuTER/ULM*, controlled by an on-board industrial PC.

We point out that in this work, we only utilize the *RobuTER* mobile base; the *ULM* manipulator is not considered.

5.2 Description of the validation scenario

The validation of the proposed *RRT-PCHIP* planning approach is carried out through the same scenario established in (Bakdi et al., 2017). The differential mobile robot, *RobuTER*, evolves in the cluttered indoor workspace described in Fig. 10. The robot has to move from the initial position $Source(x, y) = q(t = 0) = (74mm, 1953mm)$ toward the final position $Target(x, y) = q(t = T_{Transfer}) = (5501mm, 1308mm)$ with a maximum linear speed of $v_{max} = 150mm/s$

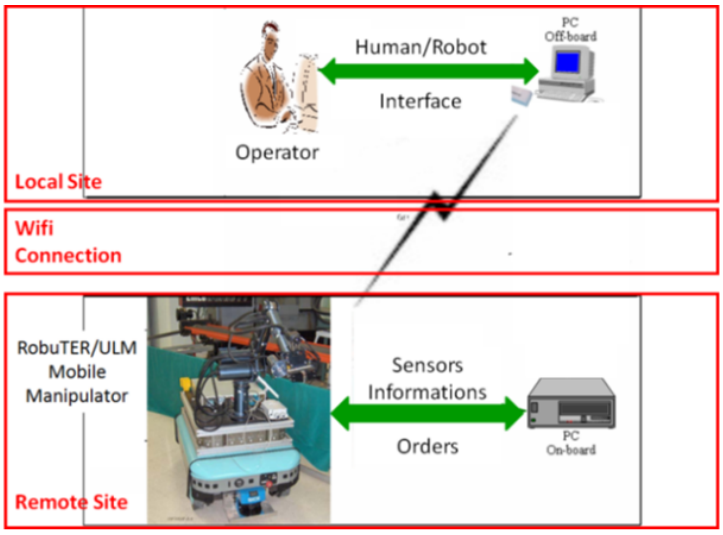


Figure 9: Architecture of the experimental robotic system.

Figure 10: Cluttered environment where the robot evolves (Concatenation of the two images captured by Kinect 1 and Kinect 2).

($v_{min} = -150mm/s$). In addition, the boundary velocities are fixed to null (i.e., $v(t = 0) = v(t = T_{Transfer}) = 0$). As it can be seen, the initial and final orientations of the mobile robot (θ_B) are not taken into account when dealing with this scenario.
The appropriate right (V_R) and left (V_L) velocities are calculated before motion begins on the host PC. A wireless communication is established between this PC and the on-board PC of the *RobuTER* mobile robot to send velocities, and to receive the odometry readings which are used for feedback control of the robot.

5.3 Obtained results

RRT chooses the first feasible path and creates its nodes randomly. Consequently, the algorithm is executed many times with different step sizes; the best path referring to the (*i*) *generation time*, (*ii*) *length of the path*, (*iii*) *number of segments* and (*iv*) *transfer time* is selected. The chosen distance for the region test is $DIST = 20\ pixels$. In addition, eight main step sizes (in

pixels) have been randomly chosen: 8, 10, 20, 30, 35, 40, 50 and 60 which correspond to (in mm) 50.72, 63.4, 126.8, 190.2, 221.9, 253.6, 317 and 380.4, respectively.
Table 2 summarizes the average of the obtained results after 10 runs for each step size. We can clearly see that executing the *RRT* with $STEP = 40\ pixels$ (i.e., $253.6\ mm$) presents the best compromise comparing with the other step sizes. The resulted path has been generated in $T_{Generation} = 0.81\ seconds$, its length is $\ell = 6401.00\ mm$, it contains 10 segments ($Edges\ number = 10$) and the transfer time is $T_{Transfer} = 40.53\ seconds$. The obtained path

Table 2: Comparison between the obtained results for different $STEP$ sizes.

$STEP$ **size (pixels)**	**Generation time (s)**	**Path length (mm)**	**Number of edges**	**Transfer time (s)**
08	10.00	7303.68	44	48.83
10	03.02	6632.60	32	42.57
20	01.58	6798.20	18	44.25
30	01.06	6703.30	13	42.02
35	01.00	7100.80	12	47.83
40	00.81	6401.00	10	40.53
50	00.63	6764.00	09	41.56
60	00.69	6284.80	07	41.22

with $STEP = 40\ pixels$ is shown in Fig. 11. The orange lines represent the generated random tree G. The purple segments correspond to the returned path before reducing the number of nodes; the pink segments represent the final path after reduction of nodes number. As it can be observed on this figure, the generated paths are zigzag segments for all the step sizes. Therefore, to get more efficient results, *PCHIP* has been used to smooth the found path; the final result is shown by the red curve in Fig. 12. The found collision-free path must be carried

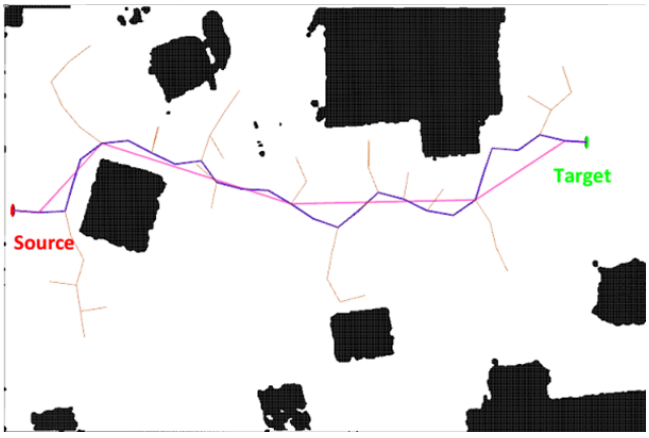


Figure 11: *RRT* planning with a step size of 40 pixels visualized in *RVIZ*.

out by the differential mobile robot; hence, $V_R(t)$ and $V_L(t)$ have to be calculated. The variations of the right wheel, left wheel and linear velocities for path tracking by the *RobuTER* robot are shown in Fig. 13. From Fig. 13, it can be seen that the maximum linear speed of the mobile robot while performing the generated path is $v_{max} = 150\ mm/s$; acceleration from $t = 0\ second$

Figure 12: *RRT-PCHIP* smoothed path linking $Source$ to $Target$ visualized in *RVIZ*.

to $t = 2\ seconds$, constant speed from $t = 2\ seconds$ to $t = 38\ seconds$ and, finally, deceleration until $t = 40.53\ seconds$. At the first turning of the robot (to the left) at about $t = 2\ seconds$ (Fig.

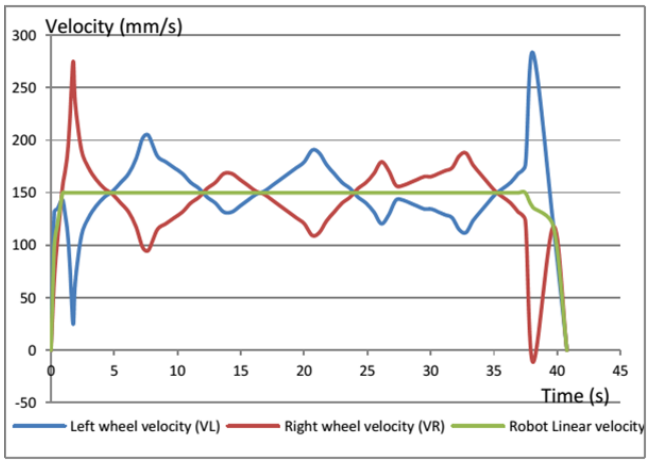


Figure 13: Variation of the robot velocities (V_L, V_R and V_{Linear}) for path tracking.

12), the right and left velocities of the robot change excessively. This is due to the significance of the turning angle that the robot must initiate. For the second turning at about $t = 7\ seconds$ (Fig. 12), the robot has to go right; however, the turning angle is not as significant as the precedent. The same phenomenon occurs at $t = 22\ seconds$ and $t = 33\ seconds$ where the robot needs to go right and left, respectively. At the last turning of the robot (to the left) at about $t = 38\ seconds$ (Fig. 12), the right and left velocities of the robot change greatly before becoming null at $t = 40.53\ seconds$ (Fig. 13).

5.4 Comparison

We have mentioned that the same scenario (same environment and same conditions) has been carried out using *GA-PCHIP* approach in (Bakdi et al., 2017). To confirm the effectiveness and superiority of our approach, Table 3 shows a comparison between the results of executing these two algorithms (*RRT-PCHIP* and *GA-PCHIP*) on the same environment in terms of (*i*) *generation time*, (*ii*) *length of the path*, (*iii*) *number of segments* and finally, (*iv*) *transfer time*.

It must be noted here that the considered final results for the *GA-PCHIP* approach are those for generating a feasible path (not for the optimal path).
From Fig. 12, *RRT-PCHIP* avoids the first obstacle from the left; on the other hand, the *GA-PCHIP* did this from the right. This can be justified by the purposes of both approaches. Indeed, the main difference between the two methodologies remains in the fact that *GA-PCHIP* searches for the optimal path (in terms of its objective function); whereas, *RRT-PCHIP* only seeks for a feasible path (without collision). From Table 3, in order to generate the path connecting $Source$ to $Target$, our approach is approximately six times faster than *GA-PCHIP* approach. Second, we noted that *GA-PCHIP* takes longer distance to move between these two configurations comparing with *RRT-PCHIP*. Third, the proposed *RRT-PCHIP* algorithm generates a path with less number of segments than *GA-PCHIP*. Finally, the transfer time given by the proposed approach is better than that of the *GA-PCHIP*. All these differences can simply be justified by the objective function to be optimized by the *GA-PCHIP* (length, total deviation and collision avoidance).

Table 3: Summary of comparative analysis between *RRT-PCHIP* and *GA-PCHIP*.

Parameters	**RRT-PCHIP**	**GA-PCHIP**
Generation time (s)	0.81	6.33
Path length (mm)	6401.00	6587.76
Number of segments	10	18
Transfer time (s)	40.53	43.91

Another comparison is done regarding the calculation times of generating feasible paths for different workspaces with different dimensions and number of obstacles. For each case, we considered an array of Kinect cameras with various $Source - Target$ positions. Table 4 and Fig. 14 summarize the average for 10 different runs of *RRT-PCHIP* and *GA-PCHIP* algorithms (without path optimization and path execution). From Table 4, it is clear that *RRT-PCHIP* is better and more efficient compared with *GA-PCHIP* proposed in (Bakdi et al., 2017). The

Table 4: Summary of the average calculation times of 10 different runs with different number of Kinect cameras and $Source - Target$ positions.

Resolution	**Number of obstacles**	**Generation time of a feasible path (s)**	
		RRT-PCHIP	**GA-PCHIP**
640*480 (01 Kinect camera)	**05**	0.80	05.09
	10	1.04	05.59
	15	1.25	06.46
640*960 (02 Kinect camera)	**10**	1.54	06.33
	15	1.78	08.03
	20	2.11	08.47
640*1280 (04 Kinect camera)	**15**	3.03	09.62
	20	3.20	10.60
	25	3.51	10.96

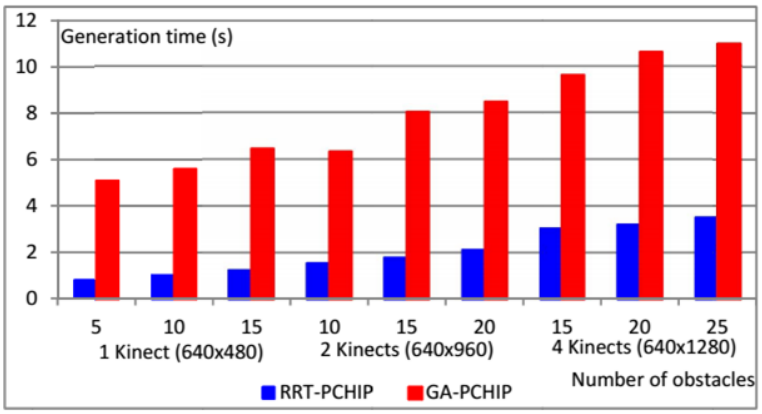


Figure 14: Average of generation times for 10 runs with different number of obstacles and $Source - Target$ positions.

generation time of feasible paths using *RRT-PCHIP* in the different environments is around three times faster than *GA-PCHIP*.

6 Conclusions and future works

This paper presented an off-line Kinect-based collision-free path planning for autonomous mobile robot using the *RRT-PCHIP* algorithm. The robot is able to successfully achieve several tasks including (*i*) perceiving and modeling its surrounding environment and (*ii*) planning its collision-free smoothed path. *ROS* is used as a framework to implement the proposed strategy, while *RVIZ* is used for visualizing the obtained results. *RRT* and *PCHIP* have been implemented to rapidly generate a feasible smoothed path connecting $Source$ to $Target$, ensuring that the robot will safely move between these two configurations. Comparisons with similar works of the literature (same environment and conditions) using *GA-PCHIP* have been done. Obtained results confirm that *RRT-PCHIP* is better than *GA-PCHIP* in terms of (*i*) *generation time*, (*ii*) *length of the path*, (*iii*) *number of segments* and (*iv*) *transfer time*.
The results reported in this article can be extended in a number of directions. The developed *RRT-PCHIP* approach will be extended first to deal with optimal path planning problem. After that, it will be validated in more complex and dynamic environments. Finally, the developed *RRT-PCHIP* will be extended to deal with on-line path planning problems for non-static environments with larger workspaces.

References

Adiyatov, O. and Varol, H. A. 2013. Rapidly-exploring random tree based memory efficient motion planning, *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, IEEE, pp. 354–359.

Almayyahi, A., Wang, W., Hussein, A. A. and Birch, P. 2017. Motion control design for unmanned ground vehicle in dynamic environment using intelligent controller, *International Journal of Intelligent Computing and Cybernetics* **10**(4): 530–548.

Ayawli, B. B. K., Chellali, R., Appiah, A. Y. and Kyeremeh, F. 2018. An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning, *Journal of Advanced Transportation* **2018**.

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

Burns, B. and Brock, O. 2007. Single-query motion planning with utility-guided random trees, *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, pp. 3307–3312.

Cabreira, T. M., de Aguiar, M. S. and Dimuro, G. P. 2013. An extended evolutionary learning approach for multiple robot path planning in a multi-agent environment, *Evolutionary computation (cec), 2013 IEEE congress on*, IEEE, pp. 3363–3370.

Cabreira, T. M., Dimuro, G. P. and de Aguiar, M. S. 2012. An evolutionary learning approach for robot path planning with fuzzy obstacle detection and avoidance in a multi-agent environment, *Social simulation (bwss), 2012 third brazilian workshop on*, IEEE, pp. 60–67.

Canny, J. 1988. *The complexity of robot motion planning*, MIT press.

Das, P., Behera, H., Jena, P. and Panigrahi, B. 2016. Multi-robot path planning in a dynamic environment using improved gravitational search algorithm, *Journal of Electrical Systems and Information Technology* **3**(2): 295–313.

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

Defoort, M., Palos, J., Kokosy, A., Floquet, T. and Perruquetti, W. 2009. Performance-based reactive navigation for non-holonomic mobile robots, *Robotica* **27**(2): 281–290.

Dhariwal, A., Sukhatme, G. S. and Requicha, A. A. 2004. Bacterium-inspired robots for environmental monitoring, *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 2, IEEE, pp. 1436–1443.

Fahimi, F. 2008. *Autonomous robots: modeling, path planning, and control*, Vol. 107, Springer Science & Business Media.

Garcia, I. and How, J. P. 2005. Improving the efficiency of rapidly-exploring random trees using a potential function planner, *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, IEEE, pp. 7965–7970.

Haidegger, T., Kovács, L., Precup, R.-E., Benyó, B., Benyó, Z. and Preitl, S. 2012. Simulation and control for telerobots in space medicine, *Acta Astronautica* **81**(1): 390–402.

Hentout, A., Benbouali, M. R., Akli, I., Bouzouia, B. and Melkou, L. 2013. A telerobotic human/robot interface for mobile manipulators: A study of human operator performance, *Control, Decision and Information Technologies (CoDIT), 2013 International Conference on*, IEEE, pp. 641–646.

Hentout, A., Bouzouia, B., Toumi, R. and Toukal, Z. 2009. Agent-based coordinated control of mobile manipulators, *The International Conference on Systems and Processing Information (ICSIP'09)*, University of Guelma, Algeria.

Hentout, A., Lehtihet, H., Chettibi, T. and Bouzouia, B. 2010. Roadmap-based collision-free trajectory planning for manipulator robots, *Journal of Modelling & Simulation of Systems* **1**(1): 40–49.

Hossain, M. A. and Ferdous, I. 2015. Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, *Robotics and Autonomous Systems* **64**: 137–141.

Hyman, J. M. 1983. Accurate monotonicity preserving cubic interpolation, *SIAM Journal on Scientific and Statistical Computing* **4**(4): 645–654.

Ibraheem, I. K. and Ajeil, F. H. 2018. 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* .

Jordan, M. and Perez, A. 2013. Optimal bidirectional rapidly-exploring random trees.

Kagami, S., Kuffner, J., Nishiwaki, K., Okada, K., Inaba, M. and Inoue, H. 2003. Humanoid arm motion planning using stereo vision and rrt search, *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Vol. 3, IEEE, pp. 2167–2172.

Kala, R. 2013. Rapidly exploring random graphs: motion planning of multiple mobile robots, *Advanced Robotics* **27**(14): 1113–1122.

Kang, L., Zhao, C.-X. and Guo, J.-H. 2009. Improved path planning based on rapidly-exploring random tree for mobile robot in unknown environment [j], *Pattern Recognition and Artificial Intelligence* **3**.

Karaman, S. and Frazzoli, E. 2009. Sampling-based motion planning with deterministic μ-calculus specifications, *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, IEEE, pp. 2222–2229.

Karaman, S. and Frazzoli, E. 2010. Incremental sampling-based algorithms for optimal motion planning, *Robotics Science and Systems VI* **104**: 2.

Karaman, S. and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning, *The international journal of robotics research* **30**(7): 846–894.

Kavraki, L. E., Svestka, P., Latombe, J.-C. and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE transactions on Robotics and Automation* **12**(4): 566–580.

Kavraki, L. and Latombe, J.-C. 1994. Randomized preprocessing of configuration for fast path planning, *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, IEEE, pp. 2138–2145.

Knispel, L. and Matousek, R. 2013. A performance comparison of rapidly-exploring random tree and dijkstras algorithm for holonomic robot path planning, *Institute of Automation and Computer Science, Faculty of Mechanical Engineerig, Brno University of Technology* .

Kothari, M., Postlethwaite, I. and Gu, D.-W. 2010. A suboptimal path planning algorithm using rapidly-exploring random trees., *International Journal of Aerospace Innovations* **2**.

Kuffner, J. J. and LaValle, S. M. 2000. Rrt-connect: An efficient approach to single-query path planning, *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, Vol. 2, IEEE, pp. 995–1001.

Kuwata, Y., Fiore, G. A., Teo, J., Frazzoli, E. and How, J. P. 2008. Motion planning for urban driving using rrt, *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, pp. 1681–1686.

LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning.

LaValle, S. M. 2006. *Planning algorithms*, Cambridge university press.

LaValle, S. M. and Kuffner Jr, J. J. 2001. Randomized kinodynamic planning, *The international journal of robotics research* **20**(5): 378–400.

Li, C., Bodkin, B. and Lancaster, J. 2009. Programming khepera ii robot for autonomous navigation and exploration using the hybrid architecture, *Proceedings of the 47th Annual Southeast Regional Conference*, ACM, p. 31.

Lindemann, S. R. and LaValle, S. M. 2004. Incrementally reducing dispersion by increasing voronoi bias in rrts, *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 4, IEEE, pp. 3251–3257.

Ma, Y., Wang, H., Xie, Y. and Guo, M. 2014. Path planning for multiple mobile robots under double-warehouse, *Information Sciences* **278**: 357–379.

Ma, Y., Zamirian, M., Yang, Y., Xu, Y. and Zhang, J. 2013. Path planning for mobile objects in four-dimension based on particle swarm optimization method with penalty function, *Mathematical Problems in Engineering* **2013**.

Ma, Y., Zheng, G., Perruquetti, W. and Qiu, Z. 2015. Local path planning for mobile robots based on intermediate objectives, *Robotica* **33**(4): 1017–1031.

Maoudj, A., Hentout, A., Bouzouia, B. and Toumi, R. 2018. On-line fault-tolerant fuzzy-based path planning and obstacles avoidance approach for manipulator robots, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **26**(5): 809–839.

Neto, A. A., Macharet, D. G. and Campos, M. F. 2018. Multi-agent rapidly-exploring pseudo-random tree, *Journal of Intelligent & Robotic Systems* **89**(1-2): 69–85.

Nieto, J., Slawinski, E., Mut, V. and Wagner, B. 2010. Online path planning based on rapidly-exploring random trees, *Industrial Technology (ICIT), 2010 IEEE International Conference on*, IEEE, pp. 1451–1456.

Noreen, I., Khan, A. and Habib, Z. 2016. A comparison of rrt, rrt* and rrt*-smart path planning algorithms, *International Journal of Computer Science and Network Security (IJCSNS)* **16**(10): 20.

Pandey, A. and Parhi, D. R. 2016. Autonomous mobile robot navigation in cluttered environment using hybrid takagi-sugeno fuzzy model and simulated annealing algorithm controller, *World Journal of Engineering* **13**(5): 431–440.

Pepy, R. and Lambert, A. 2006. Safe path planning in an uncertain-configuration space using rrt, *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, pp. 5376–5381.

Purcaru, C., Precup, R.-E., Iercan, D., Fedorovici, L.-O. and David, R.-C. 2013. Hybrid pso-gsa robot path planning algorithm in static environments with danger zones, *System theory, control and computing (ICSTCC), 2013 17th international conference*, IEEE, pp. 434–439.

Qureshi, A. H. and Ayaz, Y. 2015. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments, *Robotics and Autonomous Systems* **68**: 1–11.

Rath, M. K. and Deepak, B. 2015. Pso based system architecture for path planning of mobile robot in dynamic environment, *Communication Technologies (GCCT), 2015 Global Conference on*, IEEE, pp. 797–801.

ROS 2018. `http://www.ros.org/`.

Sgorbissa, A. and Verda, D. 2013. Structure-based object representation and classification in mobile robotics through a microsoft kinect, *Robotics and Autonomous Systems* **61**(12): 1665–1679.

Soriano, Á., Bernabeu, E. J., Valera, Á. and Vallés, M. 2014. Distributed collision avoidance method based on consensus among mobile robotic agents, *International Journal of Imaging and Robotics* **15**(1): 80–90.

Su, K., Wang, Y. and Hu, X. 2015. Robot path planning based on random coding particle swarm optimization, *International Journal of Advanced Computer Science and Applications* **6**(4): 58–64.

Sureshkumar, W., Mahalingam, K. and Rama, R. 2017. Robot motion planning inside a grid using membrane computing, *International Journal of Imaging and Robotics* **17**(1): 14–26.

Tzafestas, S. G. 2013. *Introduction to mobile robot control*, Elsevier.

Vonásek, V., Faigl, J., Krajník, T. and Přeučil, L. 2009. Rrt-path–a guided rapidly exploring random tree, *Robot Motion and Control 2009*, Springer, pp. 307–316.

Vrkalovic, S., Teban, T.-A. and Borlea, I.-D. 2017. Stable takagi-sugeno fuzzy control designed by optimization, *International Journal of Artificial Intelligence* **15**(2): 17–29.

Yacoub, R. R., Bambang, R. T., Harsoyo, A. and Sarwono, J. 2014. Dsp implementation of combined fir-functional link neural network for active noise control, *International Journal of Artificial Intelligence* **12**(1): 36–47.

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

Yu, J., Kromov, V. et al. 2001. A rapid path planning algorithm of neural network, *Robot* **23**(3): 201–205.