JCTA Journal of Computing Theories and Applications ISSN:3024-9104

Research Article

NURBs Based Multi-robots Path Planning with Obstacle Avoidance

Hadjira Belaidi[1,*] and Fethi Demim[2]

[1] Signals and Systems Laboratory, Institute of Electrical and Electronic Engineering, University M'Hamed BOUGARA of Boumerdes, Boumerdès, Algeria;
e-mail : hadjira983@yahoo.fr, ha.belaidi@univ-boumerdes.dz
[2] Guidance and Navigation Laboratory, Ecole Militaire Polytechnique, Bordj El Bahri, Algiers, Algeria;
email: demifethi@gmail.com
* Corresponding Author : Hadjira Belaidi

Abstract: The primary problem for multi-robot displacement and motion phase solving requires that the robots prevent themselves from colliding with each other as well as stationary obstacles. In certain situations, robot conflict is unavoidable if one robot views its neighbors as immovable obstacles. Hence, this paper proposes a new NURBs (Non-Uniform Rational B-spline) based algorithm for multi-robot path planning in a crowded environment. First, the proposed technique finds each robot's free, smooth, optimal path while avoiding collision with the existing obstacles. Secondly, the prospect of possible collision between the preplanned trajectories will be computed to allow the robots to navigate in the same workspace and coordinate between them. Then, each robot's time to arrive at potential collision sites is computed based on its speed. As a result, the robots involved in the collision must choose whether to use the robot priority technique to prevent the collision. Simulation results under different scenarios and comparisons with previous works are provided to validate the work. The obtained results prove that the proposed approach is accurate (as the robot's instantaneous speed is taken into consideration), fast (as there is no need to broadcast the robots' positions), the robots' paths are optimal and smooth (to avoid jerk movements), and the approach ensures that the robots will not be trapped by local minima problem.

Keywords: Multi-robots; NURBs; Obstacle avoidance; Path planning; Robot velocity.

1. Introduction

There are several benefits that a group of parallel-operating mobile robots has over individual robot systems. When working together, several robots can complete jobs faster than a single robot and can complete tasks that one robot system cannot handle. With a wide range of applications, including autonomous warehouses[1], [2], the military, aircraft, locating things or people, exploring surroundings, and rescue, it provides the qualities of parallelism, resilience, and flexibility. The main topic of current research is the distribution of tasks and resources, sensing, resolving conflicts, formation, coordinated planning, and other issues [3], [4]. One of the main challenges in resolving conflicts at the motion stage is that the robots must prevent collisions with other robots and with stationary obstacles. In certain situations, robot conflict is unavoidable if one robot views its neighbors as immovable obstacles. In crowded multi-robot scenarios, it gets worse. Hence, in the fields of robotics, video games, and automation, path planning is an essential activity in the automation process of a system that navigates the environment while avoiding obstacles and respecting various constraints. In the case of a mobile robot's team navigating in a crowded environment, each robot has to find its free optimal path from a given starting position to a given target while avoiding the static and dynamic obstacles existing in the surroundings and avoiding collision with the other robots.

Several works have proposed different collision avoidance methods based on the information afforded by onboard sensors. The paper [5] presented a collision-free path planning

Received: March, 17th 2024
Revised: April, 25th 2024
Accepted: April, 27th 2024
Published: May, 5th 2024

algorithm based on Bluetooth-received signal strength indication for multi-robot interior environments. However, low measurement precision results from attenuating the Bluetooth signal during propagation. The article [4] proposed a laser-based approach to help prevent collisions in intricate situations with multiple robots. The suggested method takes into account the influences of the robot itself, the stationary obstacles, and other adjacent robots. However, the collision-free paths are not guaranteed as the velocities of the robots are not considered. Hence, the speed factor is introduced in [6], where a collision prevention approach is suggested for a remote operation system with several humans in charge. Conversely, optimization is not carried out. In order to deal with dynamic impediments, the Velocity Obstacle (VO) evolved in [7]. As long as the dynamic obstruction keeps the measured velocity constant, the velocity at which a collision will finally occur is represented geometrically by the VO. However, the approach did not have the ability of robot–robot detection, and the robots needed to broadcast their locations; thus, results from speeding up the simulation would be erroneous.

In the same context of multi-robot trajectory planning, algorithms based on Artificial Potential Field (APF) reinforced by other techniques are proposed by several authors, such as Bacterial Potential Field [8] and APF method combined with the Virtual Obstacles approaches [9] to address the issue of multiple robots cooperating in within a single workspace and tried to avoid local minima problem [10]–[14]. However, the problem of these approaches is associated with scalability. Given current results, more research is necessary to address multiple robot collision avoidance, particularly in crowded areas [4], [15]. Moreover, the discussed approaches do not provide smoother trajectories, which reflect the existence of jerk movement.

Hence, this paper proposes a new NURBs (Non-Uniform Rational B-spline) based approach for multi-robots path planning in a crowded environment. The employment of NURBs is justified by several benefits, including (i) producing a smooth curve between control points; (ii) Coordinate points, which are the starting point, control point, and target point, are specified to generate the NURBS curve; (iii) The robot can turn without stopping as it guarantees the C^2 continuance; (iv) The tolerance for the NURBs curve's fitting can be adjusted; (v), a fit point (or other control points) and refit the NURBs through the new set of points can be inserted without altering the entire path; (vi) The robot's initial and ending rotations can be used to determine the start and end tangents. First, the proposed technique finds each robot's free, smooth, optimal path while avoiding collision with the existing obstacles. Our mobile robot path planning strategy based on NURBS, previously published in [15], [16], is used to plan a free optimal path for multi-robots. However, this work will generalize the approach for multi-robots while avoiding collisions with obstacles and between the robots themselves.

Our contribution is: (i) To consider the robot's instantaneous speed to enhance the accuracy of the approach. (ii) By calculating the robots' possible collision time, there will be no need to broadcast the robots' positions, thus, increasing the approach convergence time. (iii) The obtained robots' paths must be optimal and smooth to avoid the jerk robots' movements (iv) Moreover, the proposed approach must ensure that the local minimum problem will not trap the robots. Furthermore, the robots involved in the collision must decide whether to use the robot priority strategy to prevent the collision [17].

2. Related works

Different collision avoidance techniques have been developed by several authors, depending on the data provided by onboard sensors and the implemented approaches. Each technique has its advantages and drawbacks. In this section, some solved issues are summarized, and the way our approach solves the cited problems is explained in the next sections.

Compared to the existing works discussed in the introduction, more study is required to solve multiple robot collision avoidance, especially in crowded environments[4], [16]. In addition, the current studies' optimality still needs to be improved [5]–[8]. Furthermore, the techniques that have been discussed do not offer smoother trajectories that account for jerk movement[18].

The most common hard issue the previously proposed approaches face is the lo-calminima problem [10]–[14]. The works used the "non-minimum speed algorithm" to solve this problem. Actually, though, the robot can stop at any moment to prevent running into an

undesired obstacle or other robots or moving things. On the other hand, each robot's collision-free mobility plans are determined in [19] using a convex optimization. Convexity of the environment constraints is achieved by pre-calculating the potential field forces for a horizon, which are then used as external force inputs in optimization criteria. As a result, the concepts put forward by [20] and [19] as well as the other earlier approaches, appear to be restricted. As will be explained later, the local-minima problem is satisfactorily resolved in our work and doesn't need to be calculated again, even for the U-shaped static obstacle, which was a common drawback for the earlier methods such as [19] and [21]

Additionally, while robots are traveling in separate directions on the same side of a road with two lanes, they will not block each other's paths, as demonstrated by previous research in the literature [22], [23]; however, no work has been done on robot direction movement in a given environment at random. Our proposed approach, as it will be discussed in the coming section, the robots will not have any conflicts whatever are their heading directions.

3. Optimal path planning

The strategy of constructing the free paths for each robot is summarized in the flowchart given in Fig. 1. According to this flowchart, the user has to define the robots' quantity N existing in the workspace. After that, S_i and T_i must be defined such that S_i is the initial location of the robot i and T_i is the target of the robot i depicted by the user.

The proposed path planning strategy for multi-robot is based on parametric curves defined by Equation (1), which has an inherent directional property that enormously reduces calculations in mobile robot path planning [24]. In such an application, great attention was paid to computational complexity compared with other path-planning candidates.

$$r(s) = (T_i - S_i) \times s + S_i \tag{1}$$

Where S_i and T_i $(0 < \mathrm{i} < \mathrm{N})$ are presented by their coordinates; s represents the curve parameter and $0 \leq \mathrm{s} \leq 1$. To determine if the line S_iT_i collides with an object, we must carry out an intersection checking of this line with the polygonal contours of the obstacles as explained in [24]; where the segment S_iT_i and the obstacle contour P_kP_l do not intersect unless $0 \leq \mathrm{s} \leq 1$ and $0 \leq \mathrm{t} \leq 1$.

3.1. Cost function

The initial point Si must be updated and replaced by the first optimal point $Q1$ and linked to T_i. Then, the previous procedure will be repeated to find the new local optimal control point, and so on, until all there will be no collision.

Therefore, the path segment that separates two control points is divided by the maximum velocity to define the cost function. This cost function may be calculated for each path segment for any possible path containing time units. The d_{ij}'s, or the distances between each pair of control points, are calculated from S_i to T_i Therefore, using Equation (2) (where n is the control points' number), the total distance is determined:

$$D_{feasible} = \sum_{j=1}^{n-1} d_{ij} \tag{2}$$

For every feasible path, the cost function $\boldsymbol{H_{ij}}$ is thus defined in a way that:

$$H_{ij} = \sum_{j=1}^{n-1} \frac{d_{ij}}{V_{ij}} \tag{3}$$

Since at least two possible paths can be found, the best one can be determined by assuming that the robot R_i moves at its fastest safe speed and takes the shortest route, the optimal path can be found by:

$$H_{i\,\mathrm{optimal}} = \min\{H_{ij}\} \tag{4}$$

A table ($traj\ i$) will be constructed to store all the optimal control points (Q_j) constituting the trajectory, with j is the index of the control point; $traj\ i$ will be filled according to

the flowchart given in Figure 1. Initially, S_i and T_i are stored inside as the trajectory's beginning and end positions, respectively. Then, each time a new control point is found it will be inserted before T_i so that T_i always remains the last point until Q_j converges to T_i.

The effectiveness of this method is demonstrated by its capacity to choose the shortest and which can be traversed at high speed without requiring a costly state-space search.

Once all the optimal control points constituting the free trajectory are extracted, these points will be linked by a NURBS (Spline i) to construct the free optimal path for the robot i. Therefore, i is incremented so that this algorithm will be executed by all the robots in the team from $i = 1$ to N. Figure 1 illustrates the approach of free optimal path construction for a multi-robots team.

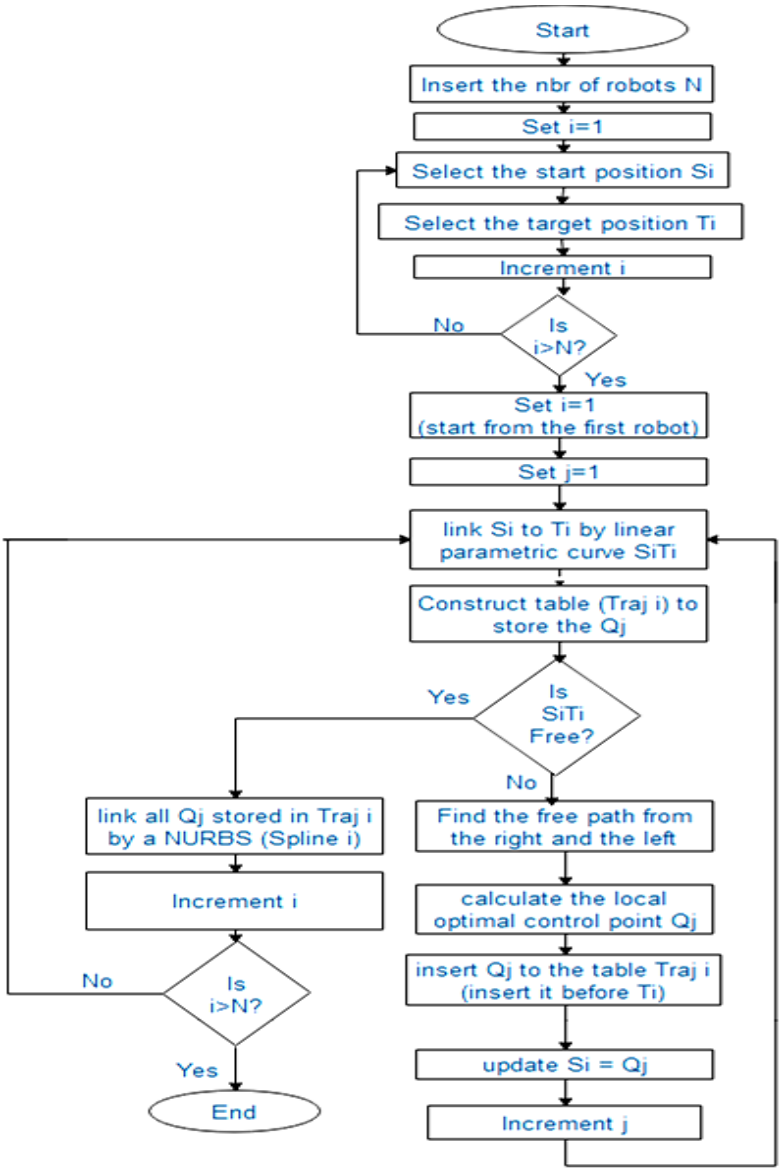


Figure. 1. Flowchart illustrating the free optimal paths construction for the multi-robots team.

4. Robot-robot Collision

Formally, the free optimal paths allow the robots to move from initial positions S_i to final destinations T_i are extracted. These paths may intersect; however, their intersection must not lead to any collision between robots. Hence, in this section, the path intersection points will be established. Then, the time required by each robot to reach these points must be calculated and compared to predict the possibility that the robots will arrive at the intersection points simultaneously, which leads to collisions.

Assuming that the robots start moving at the time t_0. The algorithm begins by checking for the intersection between the trajectory of the robot R_i, presented by Splinei, and the trajectory of a robot R_p, presented by Spline_p. The intersections between the robots trajectories are calculated as explained in the following section.

4.1. Trajectories intersection

In [17], by using the control polygon's first zero to estimate the spline's first zero, one can use the close rapport between the control polygon and the spline to identify a spline function's first zero. The novel control polygon's first zero is then utilized as an enhanced estimation, and then this zero is entered into the knot vector of the spline. Iterating over this yields an iterative technique that converges quadratically to simple zeros. The technique was naturally extended to calculate two parametric spline curves' intersections in [25]. An approximation of the intersection of the two curves was made using the initial intersection of the two control polygons. Here, "first" refers to the sequence in which things happen. Two parameter values, one for each curve, are obtained from this intersection and added as new knots to the knot vectors of the associated curves. The outcome is a straightforward iterative approach that may be repeated using the novel control polygon.

Because two curves can cross even in cases when the control polygons do not, this algorithm for calculating intersections is not infallible in and of itself. The algorithm won't function in this scenario until certain preprocessing is completed. The local convex hull attribute was used for this. More precisely, there has to be at least one pair of overlapping local convex hulls if the curves intersect. Insert the midpoints of the respective parameter intervals as new knots in the curves, concentrating on the first pair of such pairs. Theorem 11 [25] states that the algorithm will eventually result in intersecting control polygons, at least in the case of a transversal intersection, if this process is repeated. The control polygons of the two NURBS curves Spline_i and Spline_p are defined as follow:

$$\text{Spline}_i(d) = \sum_{j=0}^{n-1} R(d) q_j \tag{5}$$

$$\text{Spline}_p(g) = \sum_{j=0}^{m-1} R(g) q_j \tag{6}$$

where $R(d)$ and $R(g)$ are the rational basis function defined as follows,

$$R(d) = \frac{B_{j,k}(d) \cdot w_j}{\sum_{j=0}^{n-1} B_{j,k}(d) \cdot w_j} \tag{7}$$

$$R(g) = \frac{B_{j,k}(g) \cdot w_j}{\sum_{j=0}^{m-1} B_{j,k}(g) \cdot w_j} \tag{8}$$

Where n and m are the number of control points present in the curves Spline_i and Spline_p, respectively; k represents the NURBS degree, and q_j are the control points where each one is associated with a weight $w_j \cdot B_{j,k}(d)$and $B_{j,k}(g)$ are the B-spline blending functions. The computation of the two control polygons' intersection is simple and is well explained in [21]. The intersection Algorithm 1 is shown below.

Algorithm 1. Spline Curves Intersection Computation According to [25]

Repeat: INPUT $\text{Spline}_i(d)$, $\text{Spline}_p(g)$
OUTPUT intersections points R_{ip}
1: If $\text{Spline}_i(d)$ intersect $\text{Spline}_p(g)$Then
2: Insert first intersection of control polygons
3: Follow: If intersection is found Then
4: Return (the intersection coordinates R_{ip})
5: Stop
6: Else

```
7:                                  Go to repeat
8:                              End
9:              Else
10:                 If (two local convex hulls intersect) Then
11:                     Insert first midpoint pair
12:                     Go to follow
13:                 Else
14:                     Return (Curves do not intersect)
15:                     Stop
16:                 End
17:             End
```

The flowchart given in Figure 2 explains this algorithm (Algorithm 1), and summarizes the strategy to calculate the points R_{ip} where the trajectories of the robots intersect and the time corresponding to each robot to reach those points. Such that ti and tp represent the time when robots R_i and R_p reach the collision point R_{ip} respectively. In case of collision, robot R_i has to stop for period T before following its path.

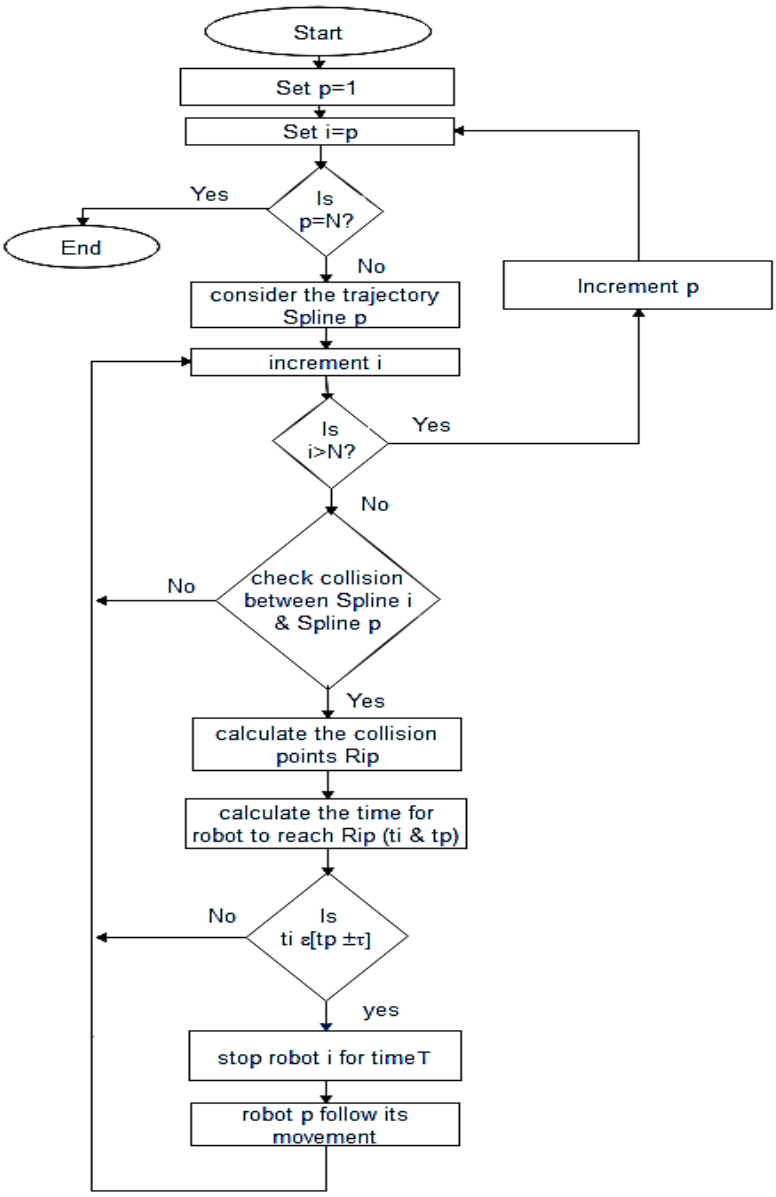


Figure. 2. Flowchart summarizing the technique of trajectories collision calculation.

4.2 Time to reach Rip by each robot

To find the time to attend the intersection points R_{ip} by each robot, the velocities along the path from the starting point until those points must be known. The speeds are denoted by V_{ij}, where i is the index of the robot, and j is the index of the control point, i.e., Q_{12} is the second control point for the first robot (R_1) and V_{12} corresponds to the speed of the first robot at the second control point. Spline_1 is the NURBS curve representing the trajectory of the first robot (R_1) and Spline2 denotes the NURBS curve characterizing the trajectory of the second robot (R_2). Whereas at the intersection point (R_{12}), the speed of the robot R_1 is VR_{12} and the speed of robot R_2 is VR_{21}. Thus, the velocities at each control point previous and at R_{ip} must be calculated.

To find the velocities V_{ip}, the first derivative of Spline_i and Spline_p at specific control points must be calculated. Spline_i and Spline_p, given by Equation (5) and (6), are written as function of the basis functions $R(d)$ and $R(g)$. These basis functions, defined by Equation (7) and Equation (8), can then be used to calculate the derivative of the curves Spline_i and Spline_p at a given point by estimating the form:

$$R(d) = \frac{w(d)R(d)}{w(d)} = \frac{A(d)}{w(d)} \tag{9}$$

$$R(g) = \frac{w(g).R(g)}{w(g)} = \frac{A(g)}{w(g)} \tag{10}$$

Their derivatives can be written as:

$$R'(d) = \frac{A'(d) - w'(d).R(d)}{w(d)} = \frac{w_j.B_{j,k}'(d) - w'(d).R(d)}{w(d)} \tag{11}$$

$$R'(g) = \frac{A'(g) - w'(g).R(g)}{w(g)} = \frac{w_j.B_{j,k}'(g) - w'(g).R(g)}{w(g)} \tag{12}$$

The derivative of a nonrational spline of degree k in relevance with a knot is another nonrational spline of degree $2k$ denoted over a knot vector calculated by the repetitive product, as the multiplication of two NURBS objects of degree k is generally an alternative NURBS object of degree $2k$ [22].

Thus, the velocities of the robot R_i at the control points, j can be found by applying Equation (13):

$$V_{ij} = \text{Spline}_i'(d) = R'(d)q_j \tag{13}$$

The same thing for a robot R_p at the control points j:

$$V_{ij} = \text{Spline}_i'(d) = R'(d)q_j \tag{14}$$

Therefore, the time t_{ij}spanned by the robot R_i to move from the control point $Q_{i(j-1)}$ to the next control point Q_{ij} can be calculated using Equation (15):

$$t_{ij} = \frac{d_{ij}}{V_{ij}} \tag{15}$$

With d_{ij} is the distance separating two adjacent control points $Q_{i(j-1)}$, Q_{ij}(the distance along a path segment). In 2D space, it can be calculated using Equation (16):

$$d_{ij} = \sqrt{(q_{ij_x} - q_{(i(j-1))_x})^2 + (q_{ij_y} - q_{(i(j-1))y})^2} \tag{16}$$

By knowing the time spanned by the robot to reach each control point; thus, the total time to reach the intersection point by the robot R_i is given by Equation (17):

$$t_i = \sum_{j=1}^{j=intersection\ point} t_{ij} \quad (17)$$

Similarly:

$$t_p = \sum_{j=1}^{j=intersection\ point} t_{pj} \quad (18)$$

The time to reach the intersection point by each robot ti and tp must be corrected according to the previous robot stops and then compared to see if the robots may collide; so the following algorithm (Algorithm 2) will be executed. Moreover, priority between the robots must be considered. Generally, robots with emergency tasks move at their maximum allowed speed; furthermore, the low-speed robot brakes more easily than the high-speed one. Hence, the robot with high speed has the priority of moving before the other robots. So, the speed of the two robots R_i and R_p , $V_i = \sum_{j=1}^{j=intersection\ point} V_{ij}$ and $V_p = \sum_{j=1}^{j=intersection\ point} V_{pj}$ respectively, are compared to indicate the one that has the priority to move and the other one has to stop (see Algorithm 2).

Algorithm 2. The Previous Robot Stops Consideration

INPUT: N_i, N_p, T
OUTPUT: t_i, t_p

Repeat: calculate ti and tp
    N_i: how many times the robot R_i has stopped
    N_p: how many times the robot R_p has stopped
    T: necessary time to escape the collision point R_{ip}
  If (R_i has stopped before) Then
    $t_i = t_i + \mathrm{T} * N_i$
    Goto compare
  Else
    Go to compare
  End if
  If (R_p has stopped before) Then
    $t_p = t_p + \mathrm{T} * N_p$
    Goto compare
  Else
    Go to compare
  End if

  compare: If $t_i \approx t_p \pm \tau$ Then
    If $V_p >= V_i$
      R_i stops at position (R_{ip-b}) for T time
      R_p follows its movement
    Else
      R_p stops at position (R_{ip-b}) for T time
      R_i follows its movement
    Else
      R_i and R_p follow their movements
    End

According to Algorithm 2, the robots R_i and R_p have to consider the time of their previous rest to avoid collision with the other robots in order to be exact in the calculation of ti and tp. Thus, T (a fixed time enough to avoid collision between the robots; it can be calculated by dividing the length of the robot supposed to be in a collision over its speed (for our case, it was fixed to 2s)), multiplied by the number of times the robot has stopped before

(N_i or N_p), which must be added to the time each robot must reach the intersection point (t_i or t_p). Therefore, the algorithm checks if t_i belongs to the interval $[t_p - \tau, t_p + \tau]$, which means that the robot R_i arrives at the point R_{ip} approximately at the same time as the robot R_p; so the two robots collide with each other. Hence, one of them has to stop at point. R_{ip-b} $((-b)$ ensures that the robot stops before entering the collision zone) and leave the other one to pass. Otherwise, the two robots pass the point R_{ip} at different times and do not collide with each other.

5. Results and Discussion

The proposed approach is tested in simulations run on an Intel Core i3 computer at 2.20 GHz with 6 GB of RAM. AutoCAD is used to build the environment scripts for the approach coding, whereas Matlab is used for speed, distance, and time calculations. In the following, some validation scenarios are proposed and discussed.

5.1 Validation Example

In this simulation example, a multi-robot team containing five robots navigates in an environment of 10x10 m². The obstacles can have any shape in C-space. The path planning approach takes into account the width of the robot. The starting locations of the robots are denoted by S_i and the targets by T_i in Figure 3. In this case, the linear robots' velocities (V_i and V_p) are supposed to be constant between each two consecutive control points, and they are 0.5m/s, 0.3 m/s, 0.6m/s, 0.4 m/s, and 0.7 m/s for robots: R_1, R_2, R_3, R_4 and R_5, respectively. The coordinates of the trajectories' intersection points and the intersection times are given in Table 1.

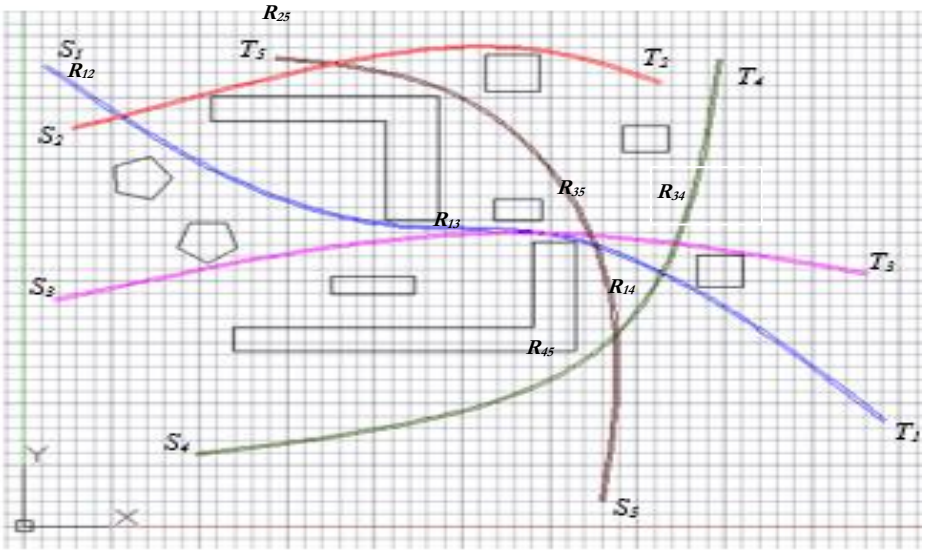


Figure. 3. Simulation illustration for path planning of five robots.

The robots' velocities are small enough to maintain the robots' stabilities during obstacle avoidance. The time to reach the intersection points is calculated for the robots where their trajectories intersect. From Table 1, the time t_1 required from robot R_1 to reach the collision point R_{12} is belongs to the interval $[t_2, t_2 + 0.5\text{s}]$, which means that the robot R_1 moving with the speed V_1 will collide with the robot R_2 moving by the speed V_2 at point R_{12}. Thus, by executing the algorithm summarized by the flowchart given in Figure 2, the robot R_1 has to stop for a short time $\mathrm{T} = 2\text{s}$ allowing $R_2$to cross the intersection point without collision. Therefore, for the next computation, T must be taken into consideration in the estimation of t_1 $(t_1 = t_1 + 2\text{s})$. The same thing to the path execution of robots R_1 and R_4; as $t_1 \epsilon [t_4 - 0.5\text{s}, t_4]$, robot R_1 has to stop for 2s, allowing the robot R_4 to pass and t_1 will be equal to $t_1 + 2 * \mathrm{T}$ (because it has stopped two times).

The proposed algorithm requires $N(N-1)/2$ computation times to detect the intersections, with N as the robots' number, and it increases as the robots' number increases. For our example, the algorithm needs to run ten times with five robots to find the corresponding collision points, as explained in Table 1.

5.2 Local minima problem

There are certain restrictions when using the APF approach for path planning, according to [20]. The main issue arises when the robot becomes stuck in nearby minima. The robot is stuck in this posture because there is zero potential field force acting on it. Similarly, repeatedly pushing the robot in the exact opposite way can result in a control deadlock [19]. Situations such as control deadlock and local minima are undesirable.

Thus, a technique based on the non-minimum speed algorithm was put forth in [20]. The idea was to regulate each robot's speed at any given moment to address the issue of local minima as they arise. The concept was to regulate the robot's force when it is still distant from the objective point, beyond a positive and non-null threshold F_{min}. Consequently, when the robot is far away from the destination, its speed is never equal to zero.

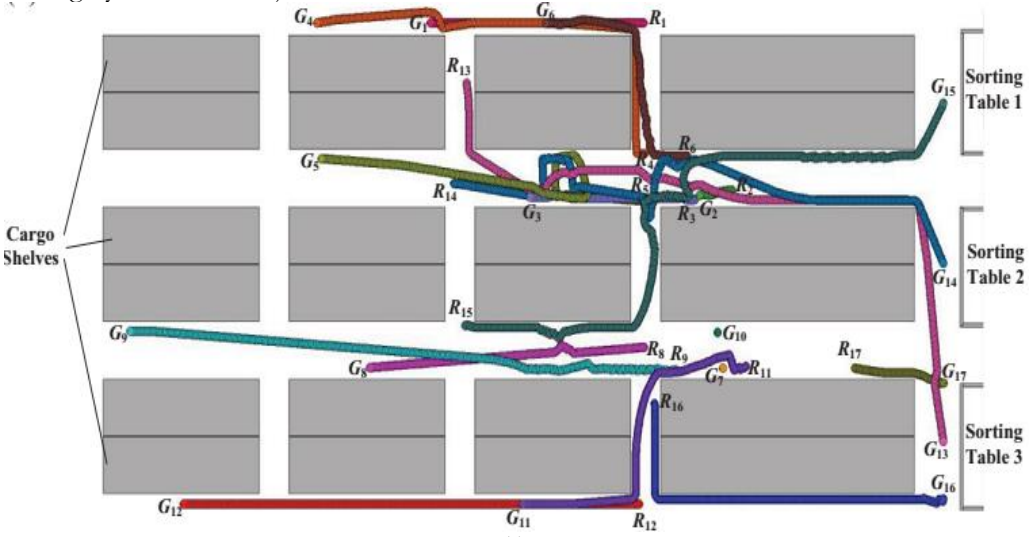
Table 1. The coordinates of the trajectories intersection points and the intersection times.

Intersection points R_{ip} (R_i with R_p)	x_{ip}	y_{ip}	dS_iR_{ip} (*10m)	dS_pR_{ip} (*10m)	V_i (m/s)	V_p (m/s)	t_i (s)	t_p (s)	Observation
R_{12}	13,56	84,23	12.78	6.41	0.5	0.3	2.56	2.14	Collision => R_1 stops for 2s and R_2 follows ($t_1 = t_1 + 2$).
R_{13}	78,20	58,96	84.13	76.81	0.5	0.6	19.32	12.80	No collision ($t_1 = t_1 + 2$).
R_{14}	90,95	51,39	99.03	79.49	0.5	0.4	22.31	21.87	Collision => R_1 stops for 0.5s and R_4 follows ($t_1 = t_1 + 2 * 2$)
R_{15}	81,44	57,87	87.55	53.51	0.5	0.7	20.51	7.64	No collision
R_{23}	No possible intersection of the robots' trajectories								
R_{24}	No possible intersection of the robots' trajectories								
R_{25}	40,79	92,46	34.85	105.51	0.3	0.7	11.61	15.07	No collision
R_{34}	93,33	57,87	90.14	86.39	0.6	0.4	15.02	21.6	No collision
R_{35}	81,00	58,74	75.85	54.44	0.6	0.7	12.64	7.78	No collision
R_{45}	84,68	39,50	66.16	34.86	0.4	0.7	16.54	4.98	No collision

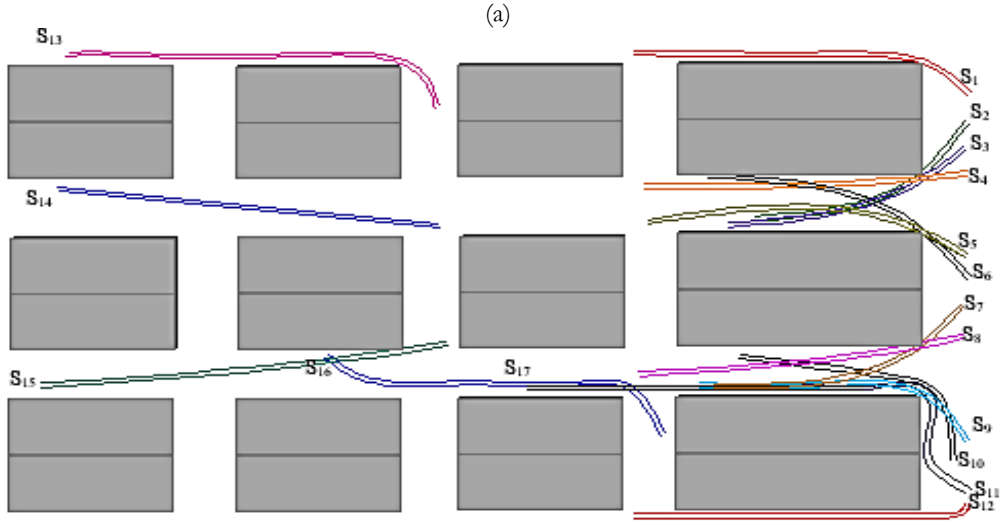
The local minima problem is resolved by the "non-minimum speed algorithm". However, the robot can stop anytime to avoid an unwanted obstacle or collision with other robots or moving objects. Whereas, in [17] a convex optimization is used to determine each robot's collision-free motion plans. A pre-computation of the potential field forces for a horizon are proposed to convexity the environment constraints and then use them as external force inputs in optimization criteria. Therefore, the ideas presented by [17] and [25], like the other previous methods, seem limited. In our work, the problem of local minima is solved successfully and does not require any more calculation, as explained below.

Hence, Figure 4 shows a comparison between the trajectories of the robots obtained by running the method proposed in [4] (Fig. 4(a and c)) and those obtained by our approach (Fig. 4(b and d)). The built scenario assumes that 17 robots navigate in a space featuring a few cargo shelves and three sorting tables. The same experience explained in [4] is re-introduced here. The initial location and the target for the robot R_q are S_q and G_q, respectively. Form the results obtained in [4], each cargo robot accomplishes collision-free navigation; however, it is clearly observed that the obtained trajectories are not optimal, and the robots' movements are not smooth, which causes them to jerk. Thus, the robot's movement is not stable. However, our approach produced an optimal path, smooth robot movements, and no oscillation in the robot's speeds. Moreover, the obtained paths are more secure even if the robots move at optimal speed.

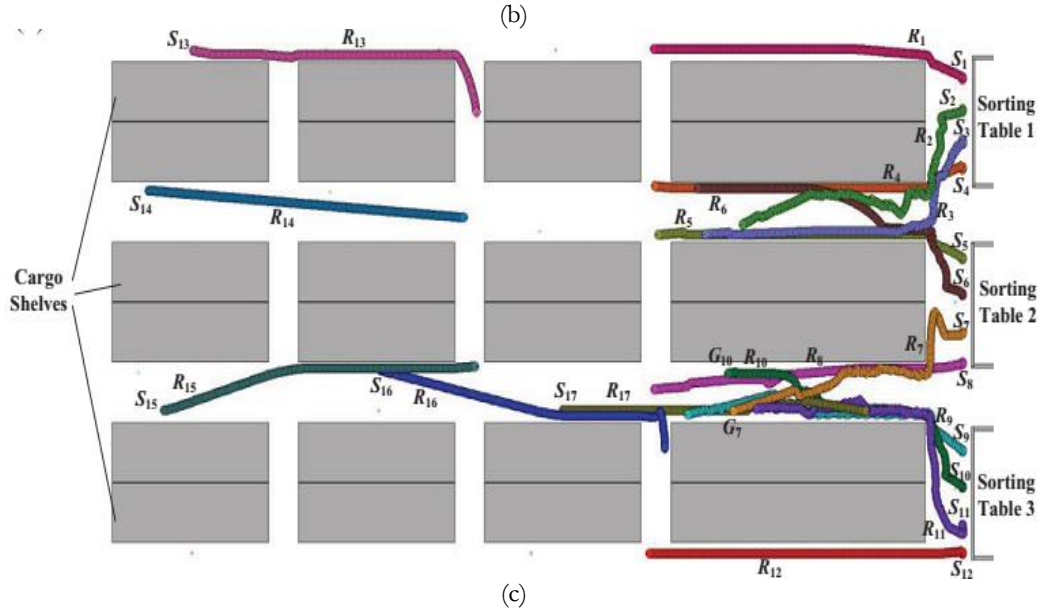
Moreover, in our proposed approach, the robots move in a straight path when there is no obstacle, which is not the case in [4]. The same improvement is proved when we compared our results with the ones obtained by the GLAS end-to-end policy approach proposed by [22], where the objective is to move robots from the initial location (circles) to the desired location (squares) as illustrated in Fig. 4(e) and Fig. 4(f). Our method achieved more optimal path highly smoothed trajectories.



(a)



(b)



(c)

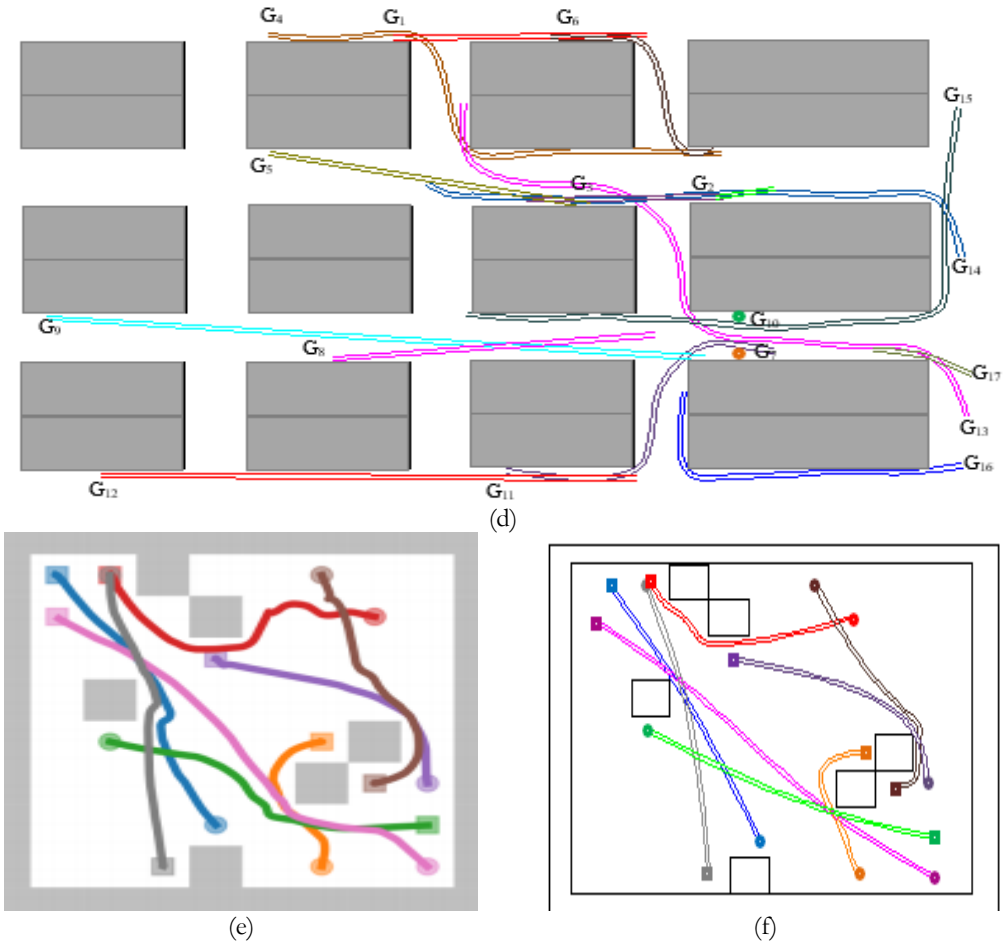


Figure 4. Comparison of our results with those obtained in [4] and [26] (**a**) and (**c**) Trajectories of the robots obtained by the approach proposed by [4]; (**b**), (**d**) and (**f**) Trajectories of the robots obtained by our approach; (**e**) Results obtained from [26].

Generally, while applying approaches such as DQMPC (Decentralized Quadratic Model Predictive Control), approach Angle Towards Target Method, and Swiveling Robot Destnation Method, etc [17], [21]. When faced with a U-shaped static obstacle, the robots become stuck due to a control deadlock. Figure 5 illustrates local minima escaping while the robots face a U-shaped obstacle using our approach. In this scenario, three robots move in a complex environment, and their targets are reached effectively.

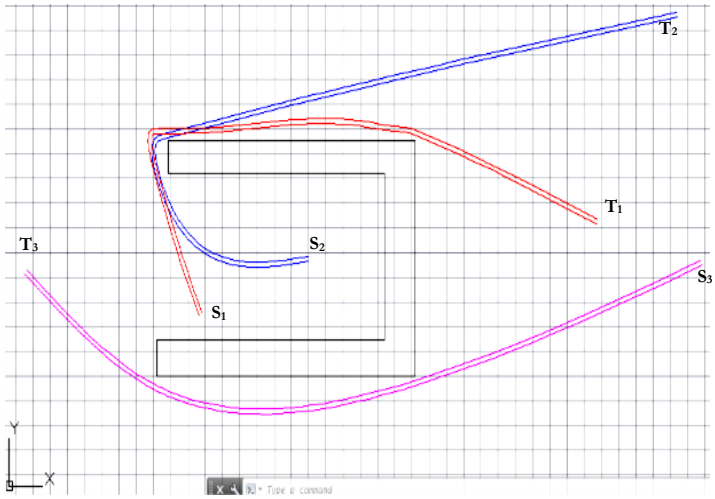


Figure 5. Escaping local minima problem caused by U-shaped obstacle.

5.3 Conflict-free paths and obstacle avoidance

The existing works in the literature proved that robots running on the same side of a two-way street would not obstruct each other's paths while they are headed in separate directions [22], [23]; however, no work has been planned for random robot movement direction in a given environment. In our work, and according to what has been proven in the previous sections (see section 2 and section 3), the planned path will not conflict with the other robots wherever it is heading. Moreover, this approach can also be applied to several target executions, where each target can be inserted as a new control point and adjust the pre-planned path according to its position.

One limitation of the proposed algorithm is that one of the robots is going out of service when it reaches one of the intersection points. This may cause an infinite increase in the robot's time stop. This situation can be solved by adding another constraint to the initial algorithm by considering the target reachability and limiting the maximum time for the task execution. So, the robot that didn't respect the maximum time for its task execution is considered out of service, and the algorithm will be reinitialized, and the number of the robots will decrease by one, considering the robot out of service as an obstacle.

Moreover, the proposed algorithm requires $N(N-1)/2$ computation times to detect the intersections, with N is the robot's number. To consider the algorithm limitation, the assumption is that $N(N-1)/2$ must be less than the requested task execution time $(N(N-1)/2 < \text{task execution time})$.

6. Conclusions

This paper proposed a new algorithm for multi-robots path planning in a crowded environment. A free smooth optimal path for each robot from its initial to its final position while avoiding collision with the existing obstacles is planned; then, the likelihood of a collision between the pre-planned paths is determined, and the amount of time it takes for each robot to arrive at a potential collision location based on its speed is discussed to investigate the likelihood of a collision. Because of this, the robots involved in the collision must decide how to prevent it by using the robot priority technique. Several simulation scenarios were tested to validate the effectiveness of our proposed approach. The results obtained, and comparisons with the existing approaches have proved the effectiveness of the proposed approach. It offers smoother trajectories that account for jerk movement, it satisfactorily resolves the local-minima problem without the need for further calculation, it avoids any kind of obstacle, even the U-shaped static obstacle, and it isn't safer from any conflicts between the robots, whatever their heading directions. The number of algorithm steps depends on the number of control points.

In future work, and in order to confirm the suggested method's efficacy, further study will entail physically validating it on several robot platforms with multiple real robots. Our perspective is to implement the proposed NURBs approach on microprocessor-embedded boards to enable real robots to navigate in 3D environments. Moreover, the method can be generalized to any crowded dynamic environment.

Author Contributions: Conceptualization: H. B. and F. D..; methodology: H.B.; software: H.B.; validation: H.B.; formal analysis: H.B.; investigation: H.B.; resources: H.B.; data curation: F.D.; writing—original draft preparation: H.B.; writing—review and editing: H.B. and F.D.; visualization: H.B.; supervision: F.D.; project administration: H.B.

Funding: This research received no external funding.

Data Availability Statement: Not applicable

Acknowledgments: The authors would like to thank the "la Direction Générale de la Recherche Scientifique et du Développement Technologique (DGRSDT)", for its financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Fast ADMM for Distributed Model Predictive Control of Cooperative Waterborne AGVs," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1406–1413, Jul. 2017, doi: 10.1109/TCST.2016.2599485.

[2] V. Digani, L. Sabattini, and C. Secchi, "A Probabilistic Eulerian Traffic Model for the Coordination of Multiple AGVs in Automatic Warehouses," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 26–32, Jan. 2016, doi: 10.1109/LRA.2015.2505646.
[3] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim, "Multi-Robot Active Sensing and Environmental Model Learning With Distributed Gaussian Process," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5905–5912, Oct. 2020, doi: 10.1109/LRA.2020.3010456.
[4] Y. Yu, Z. Wu, Z. Cao, L. Pang, L. Ren, and C. Zhou, "A laser-based multi-robot collision avoidance approach in unknown environments," *Int. J. Adv. Robot. Syst.*, vol. 15, no. 1, p. 172988141875910, Jan. 2018, doi: 10.1177/1729881418759107.
[5] P. Lijina and K. A. Nippun, "Bluetooth RSSI based collision avoidance in multirobot environment," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2016, pp. 2168–2174. doi: 10.1109/ICACCI.2016.7732373.
[6] S. E. García, E. Slawiñski, V. Mut, and F. Penizzotto, "Collision avoidance method for multi-operator multi-robot teleoperation system," *Robotica*, vol. 36, no. 1, pp. 78–95, Jan. 2018, doi: 10.1017/S0263574717000169.
[7] D. Claes and K. Tuyls, "Multi robot collision avoidance in a shared workspace," *Auton. Robots*, vol. 42, no. 8, pp. 1749–1770, Dec. 2018, doi: 10.1007/s10514-018-9726-5.
[8] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using Bacterial Potential Field for avoiding static and dynamic obstacles," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5177–5191, Jul. 2015, doi: 10.1016/j.eswa.2015.02.033.
[9] A. M. Hassan, C. M. Elias, O. M. Shehata, and E. I. Morgan, "A global integrated artificial potential field/virtual obstacles path planning algorithm for multi-robot system applications," *Int. Res. J. Eng. Technol.*, vol. 4, no. 9, pp. 1198–1204, 2017.
[10] F. Matoui, B. Boussaid, and M. N. Abdelkrim, "Distributed path planning of a multi-robot system based on the neighborhood artificial potential field approach," *Simulation*, vol. 95, no. 7, pp. 637–657, Jul. 2019, doi: 10.1177/0037549718785440.
[11] Z. Wu, W. Su, and J. Li, "Multi-robot path planning based on improved artificial potential field and B-spline curve optimization," in *2019 Chinese Control Conference (CCC)*, Jul. 2019, pp. 4691–4696. doi: 10.23919/ChiCC.2019.8865232.
[12] S. Moon, E. Oh, and D. H. Shim, "An Integral Framework of Task Assignment and Path Planning for Multiple Unmanned Aerial Vehicles in Dynamic Environments," *J. Intell. Robot. Syst.*, vol. 70, no. 1–4, pp. 303–313, Apr. 2013, doi: 10.1007/s10846-012-9740-3.
[13] D. Liang, Z. Liu, and R. Bhamra, "Collaborative Multi-Robot Formation Control and Global Path Optimization," *Appl. Sci.*, vol. 12, no. 14, p. 7046, Jul. 2022, doi: 10.3390/app12147046.
[14] F. Metoui, B. Boussaid, and M. N. Abdelkrim, "Path Planning for a Multi-robot System with Decentralized Control Architecture," in *Studies in Systems, Decision and Control*, 2020, pp. 229–259. doi: 10.1007/978-981-15-1819-5_12.
[15] S.-K. Huang, W.-J. Wang, and C.-H. Sun, "A Path Planning Strategy for Multi-Robot Moving with Path-Priority Order Based on a Generalized Voronoi Diagram," *Appl. Sci.*, vol. 11, no. 20, p. 9650, Oct. 2021, doi: 10.3390/app11209650.
[16] H. Belaidi, H. Bentarzi, A. Belaidi, and A. Hentout, "Terrain Traversability and Optimal Path Planning in 3D Uneven Environment for an Autonomous Mobile Robot," *Arab. J. Sci. Eng.*, vol. 39, no. 11, pp. 8371–8381, Nov. 2014, doi: 10.1007/s13369-014-1352-8.
[17] K. Mørken and M. Reimers, "An unconditionally convergent method for computing zeros of splines and polynomials," *Math. Comput.*, vol. 76, no. 258, pp. 845–865, Jan. 2007, doi: 10.1090/S0025-5718-07-01923-0.
[18] F. Demim *et al.*, "Smooth Sliding Mode Control Based Technique of an Autonomous Underwater Vehicle Based Localization Using Obstacle Avoidance Strategy," in *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics*, 2023, pp. 529–537. doi: 10.5220/0012118200003543.
[19] R. Tallamraju, S. Rajappa, M. J. Black, K. Karlapalem, and A. Ahmad, "Decentralized MPC based Obstacle Avoidance for Multi-Robot Target Tracking Scenarios," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Aug. 2018, pp. 1–8. doi: 10.1109/SSRR.2018.8468655.
[20] F. Matoui, B. Boussaid, and M. N. Abdelkrim, "Local minimum solution for the potential field method in multiple robot motion planning task," in *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, Dec. 2015, pp. 452–457. doi: 10.1109/STA.2015.7505223.
[21] H. Belaidi, M. Abad, and A. E. Bouhamidi, "Hybrid Decentralized Multi-Robots Navigation Strategy," *J. Manuf. Technol. Res.*, vol. 13, no. 1/2, pp. 13–27, 2021.
[22] J. Gregoire, M. Čáp, and E. Frazzoli, "Locally-optimal multi-robot navigation under delaying disturbances using homotopy constraints," *Auton. Robots*, vol. 42, no. 4, pp. 895–907, Apr. 2018, doi: 10.1007/s10514-017-9673-6.
[23] J. Wang, R. Tai, and J. Xu, "A Bi-Level Probabilistic Path Planning Algorithm for Multiple Robots with Motion Uncertainty," *Complexity*, vol. 2020, pp. 1–16, Jun. 2020, doi: 10.1155/2020/9207324.
[24] 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*, Jun. 2014, pp. 168–173. doi: 10.1109/CYBER.2014.6917455.
[25] K. Mørken, M. Reimers, and C. Schulz, "Computing intersections of planar spline curves using knot insertion," *Comput. Aided Geom. Des.*, vol. 26, no. 3, pp. 351–366, Mar. 2009, doi: 10.1016/j.cagd.2008.07.005.
[26] B. Riviere, W. Honig, Y. Yue, and S.-J. Chung, "GLAS: Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning With End-to-End Learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4249–4256, Jul. 2020, doi: 10.1109/LRA.2020.2994035.