Extended Artificial Potential Field Applied for Soccer Robot Players Path Planning

1st Kian Behzad
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
kian.behzad@gmail.com

2nd Heidar Ali Talebi
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
alit@aut.ac.ir

3rd Iman Sharifi
Department of Electrical Engineering
Amirkabir University of Technology
Tehran, Iran
imansharifi@aut.ac.ir

***Abstract*—Mobile robots have recently attracted the attention of researchers due to their various potential in civilian applications. The essential technology needed for mobile robots is their navigation and path-planning. The aim of this paper is to carry out a comprehensive study on the Artificial Potential Field (APF) algorithm. This paper presents a straightforward and reliable path planning method for general mobile robots, focusing on Omni-directional soccer robot players. The (APF) method has been widely applied in static real-time path planning. In this study, we present the Extended APF (EAPF) method to solve some inherent shortcomings, such as the latency and inertia of the robots, local minima, and the non-reachability of the target (GNRON problem). We also propose a novel approach to use APF in highly dynamic environments. We added gaussian noise to the simulation's data to further resemble the real-world environment, and then we proposed a simple filter to mitigate the incoming noise as well as estimating other necessary information from the environment's raw data. Simulation results show that the proposed method has promising feasibility and efficiency in real-time path planning and can effectively avoid obstacles.**

***Index Terms*—Soccer Robot Players, Mobile Robots, Path Planning, Artificial Potential Field**

I. Introduction

mobile robot refers to a robot capable of moving in the surrounding by remote or autonomous control. They can operate autonomously where human beings cannot reach [1] and have a profound value of research.

When moving around, mobile robots will come across both static and dynamic surroundings, which means the obstacles may be stationary, moving, or both. The robots must be able to find an efficient and safe path to avoid collision. Therefore the intelligent path planning is considered to be a key technology in these researches [2]. Path planning refers to the fact that the robot finds a collision-free optimal path from the start point to the goal point in the workspace with obstacles. This optimal path could be on the premise of the one or more optimization criteria among the shortest moving path, the shortest moving time, and the minimum working cost [3] [4] [5]. Numerous algorithms for robot path planning have been developed. The primary navigation algorithms can be broadly classified into two main approaches [2]: classical and heuristics algorithms [6]. The classical approaches include roadmap building [7], cell decomposition [8], reactive approaches, and artificial potential field (APF) method.

Heuristics algorithms contain A* algorithm [9], Dijkstra algorithm [10], BFS algorithm [11], Bio-inspired algorithms [12], and etc. These algorithms can yield reasonable solutions but are not necessarily the optimum. All the mentioned algorithms have their respective advantages and disadvantages. They are not mutually independent but deeply correlated to each other. In many applications, some of them can be combined to derive the most effective path planning manner [6]. The APF method is the virtual force method that Khatib first proposed in 1986 [13]. The movement of the robot in the environmental space is transformed into the virtual force field motion. The attractive force of the goal point attracts the robot to move toward it, and the repulsive force of the obstacles prevents the robot from moving into obstacles. The resultant force combines the attractive force of the goal point and the repulsive forces of all obstacles. The magnitude and direction of the resultant force determine the desirable motion of the robot. Many advantages of APF, such as its safety and reliability, simplicity, and low computational complexity, led the researchers to investigate this method and further improve it in various applications. This paper presents a simple, straightforward, and reliable path planning method focusing on Omni-directional soccer robot players. However, by calculating the single integrator model of robots and building the proposed method on that, the research is applicable to all general mobile robots. The classical APF method has some inherent shortcomings, such as the latency and inertia of the robots (which incapacitate them to react in time when avoiding an obstacle), local minima, the non-reachability of the target (GNRON), and its inability to work under dynamic environments.

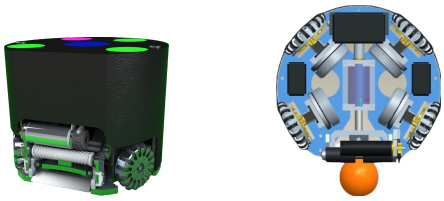
In this paper, we propose a novel approach to address all these shortcomings. We use a distance correction factor first proposed by [2] in the APF's repulsive force to solve the GNRON problem. The vertical-escape method is proposed to avoid local minima. We also introduce a trajectory-prediction approach to solve the agents' latency and inertia as well as solving the dynamic environment problem. The proposed method is tested and simulated using grSim [14]. We added gaussian noise to the grSim data to further simulate the real-world environment, and then we proposed a simple filter to mitigate the incoming noise as well as estimating other necessary information from the environment's raw data.

The rest of the paper is organized as follows. Section 2 provides the background information of the robots and the environment used in the paper. Section 3 presents our filter for noise reduction. The proposed Extended APF (EAPF) is discussed in Section 4. all results are being addressed at the end of each section.[1]

II. Robots and Environment

A. Robot

Small Size League (SSL) robots designed for Robocup competitions are considered the base environment for this paper [15]. In this environment, we have standard omnidirectional robots with four wheels such as Fig. 1 that should move around autonomously and play soccer against other robots.



(a) Front [17] (b) Bottom [18]

Fig. 1. Robots used in this paper.

B. Single Integrator Model

The input of these robots is the velocity of each wheel. However, to have a more general system applicable to other mobile robots, first we convert the velocity of wheels to the robot's linear and angular velocities, As shown in Fig. 2. We can achieve this by applying an Omni-directional Jacobian matrix and converting the wheels' velocity to the robot's speed.

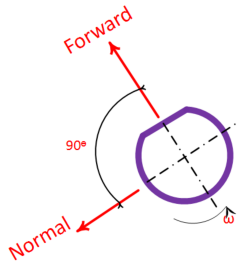


Fig. 2. Forward velocity, normal velocity, and angular velocity of robot.

After this conversion, we can generate any desirable velocity in each direction using the forward and normal velocities (orthogonal). If we assume the position of robot as the output and the its velocity as the input, our model can be described as a single integrator

$$\vec{x}(t) = \int \vec{v}(t)\, dt \tag{1}$$

where $\vec{x}(t)$ and $\vec{v}(t)$ are the position and velocity of the robot in time domain, respectively.

[1]Our code is publicly available to facilitate reproducible research: https://github.com/kianbehzad/BScThesis

C. Environment

On top of the field, we have several cameras responsible for sending images of the field in a 60Hz loop. These images are received in an image processing software called SSL-Vision, which processes the images and acquires the position of robots in the field [16]. To carry out the tests, we use the official simulator of the competitions called grSim [14], which uses the local network to send and receive information. Fig. 3 shows the simulator environment.

Fig. 3. grSim environment.

III. Filter

Data received from the robot and the environment are highly raw, only include robots' positions, and have considerable noise. This section introduces a Kalman-like filter to estimate the robot's acceleration (linear) and velocity (angular and linear) and reject the mentioned noise.

A. Velocity and Acceleration Estimation

First, we consider the noise negligible and assume that the data received from the environment are accurate. In order to estimate the linear and angular velocities from received data (positions), we can apply a first-order numeric derivative on the sequence of positions acquired from the environment as

$$\vec{v}_t = \frac{\vec{x}_t - \vec{x}_{t-1}}{\Delta t} \quad , \quad \vec{\omega}_t = \frac{\vec{\theta}_t - \vec{\theta}_{t-1}}{\Delta t} \tag{2}$$

where $\vec{x}_t$ and $\vec{\theta}_t$ are the position and angle of the robots in time t, respectively; $\vec{v}_t$ and $\vec{\omega}_t$ are also represents the linear and angular velocity at time t, and Δt is the time difference between each control loop ($60Hz$ in this research).

Remark. *When estimating the angular velocity using* (2)*, we should always consider the smaller change in the robot's angle, meaning when working with discrete images from the field, we have two separate images from one robot at two different angles. There is no possible way to determine the rotation direction of robot. Nevertheless it is more plausible to assume the robot has chosen the smaller rotate direction to go from* image 1 *to* image 2*. In order to achieve this, we can calculate* $\Delta\theta$ *using Alg.1 instead of* $\vec{\theta}_t - \vec{\theta}_{t-1}$ *in* (2)*.*

Algorithm 1 Correct calculation of $\Delta\theta$

Input $\vec{\theta}_t,\ \vec{\theta}_{t-1}$
Output $\Delta\theta_{correct}$

$\Delta\theta \leftarrow norm(\vec{\theta}_t - \vec{\theta}_{t-1})$ #angle between -180 and +180
$sign \leftarrow sign(\Delta\theta)$ # + or -
$\overline{\Delta\theta} \leftarrow \Delta\theta - 360 \times sign$ #supplementary of the $\Delta\theta$
$\Delta\theta_{correct} \leftarrow smaller(\overline{\Delta\theta}, \Delta\theta)$

Now that we have the velocity estimation, we can use it to estimate the linear acceleration of our robots by applying a first-order numeric derivative on the linear velocity as

$$\vec{a}_t = \frac{\vec{v}_t - \vec{v}_{t-1}}{\Delta t} \tag{3}$$

in which, $\vec{a}_t$ is the robot's linear acceleration in time t.

B. Noise Rejection Filter

In the absence of noise, (2) and (3) work very well. Nevertheless, the noise would cause a considerable effect on the results, especially when we have a derivative term on the data (as in (2) and (3)). To fix this problem, we will implement a Kalman-like filter and apply it to the received data. Every loop by receiving environment data, the velocity and the acceleration, will be calculated as in (2) and (3), respectively; then the outputs will be saved in a circular queue with fixed length $n-1$. Thereupon, in every loop, in addition to the currently received information, we also have access to the last $n-1$ data from previous control loops. Using each data from this queue, we could predict the current position and velocity using

$$\begin{aligned} \vec{x}_{t_{predict}} &= \vec{x}_{t-i} + (i \times \Delta t)\ \vec{v}_{t-i},\ i = 1, 2, ..., n-1 \\ \vec{v}_{t_{predict}} &= \vec{v}_{t-i} + (i \times \Delta t)\ \vec{a}_{t-i},\ i = 1, 2, ..., n-1 \end{aligned} \tag{4}$$

where the $\vec{x}_{t_{predict}}$ and $\vec{v}_{t_{predict}}$ are the position and velocity estimations at time t, respectively.
Consequently, in each loop, we have n data points for the position and velocity of robots (1 from the currently received data from the environment and $n-1$ from calculated predictions from previously saved information), and by calculating a weighted average on these n data points, we can have a confident and reliable output for position and velocity of each robot. Fig. 4 shows the workflow of the proposed filter.

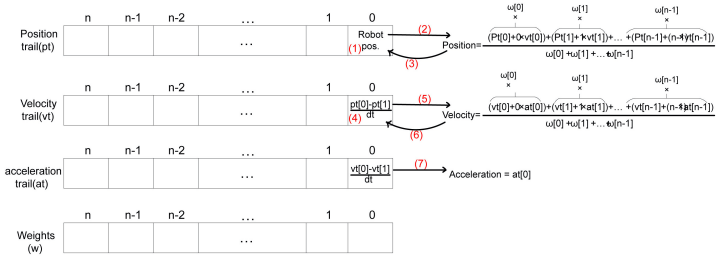


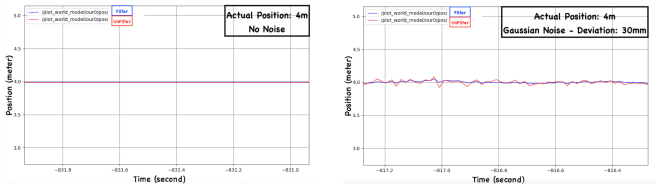
Fig. 4. Filter Workflow.

Each step of the Fig. 4 workflow will be discussed in the following.

1) Receiving the position of a robot from the environment as an initial guess.
2) Calculating the weighted average of the robot's position using predictions and (4).
3) Replacing the position's initial guess with step 2 output.
4) Estimating the robot's linear velocity using (2) as an initial guess.
5) Calculating the weighted average of the robot's velocity using predictions and (4).
6) Replacing the velocity's initial guess with step 5 output.
7) Estimating the robot's linear acceleration using (3) (needed for next loop's step 4).

C. Results

Fig. 5 shows the effect of the proposed filter on position's data in a stationary mode.



(a) No Noise (b) Gaussian Noise $\mu = 30mm$

Fig. 5. Position of a robot before and after filter.

Fig. 6 shows the effect of the proposed filter on position's data in a sinusoid move.

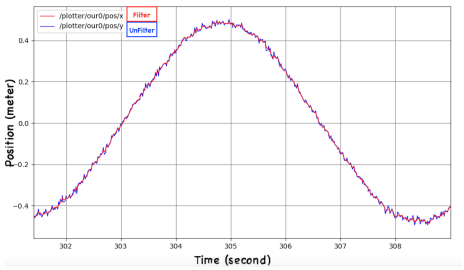
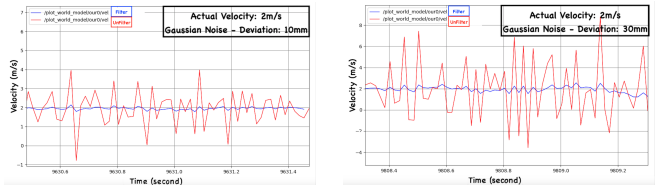


Fig. 6. Position of a robot before and after filter(Gaussian Noise $\mu = 30^{mm}$).

Fig. 7 shows the effect of the proposed filter on velocity's data in a stationary mode.



(a) Gaussian Noise $\mu = 10mm$ (b) Gaussian Noise $\mu = 30mm$

Fig. 7. Velocity of a robot before and after filter.

Fig. 8 shows the effect of the proposed filter on velocity's data in a sinusoid move.

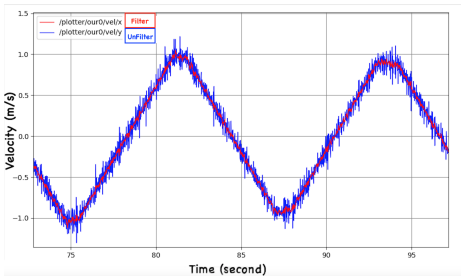


Fig. 8. Velocity of a robot before and after filter(Gaussian Noise $\mu = 10^{mm}$).

IV. Extended Artificial Potential Field

This section will implement an Extended version of Artificial Potential Field (EAPF) to control the robot's position and avoid obstacles, simultaneously. First, we introduce the APF itself, discuss its weaknesses and then introduce our extended version of APF to solve these problems.

According to the APF theory, each robot is attracted to its destination while pushes away from surrounding obstacles. If we have m number of obstacles, the resulted potential field for the i^{th} robot would be

$$U_i = U_{att,i} + \sum_{j=1}^{m} U_{rep_{ij}} \tag{5}$$

in which, $U_{att,i}$ is the attractive field generated from the robot's destination, and $U_{rep_{ij}}$ is the repulsive field generated from j^{th} obstacle to the i^{th} robot. U_i is the final and resultant field for the i^{th} robot; which we can use it to reach the destination while avoiding obstacles by moving along the opposite direction of the (5) field's gradient.

In (5), $U_{att,i}$ will be defined as

$$U_{att,i} = \begin{cases} \frac{1}{2}\,|\vec{r}_i|^2\,\zeta & |\vec{r}_i| \le d \\ d\,\zeta\,|\vec{r}_i| - \frac{1}{2}\,\zeta\,d^2 & |\vec{r}_i| > d \end{cases} \tag{6}$$

where the $\vec{r}_i$ is a vector from the destination to the i^{th} robot and will be defined using

$$\vec{r}_i = \text{robot}\vec{[i]}.\text{pos} - \text{destin}\vec{\text{ation}}, \tag{7}$$

and ζ is a parameter that indicates the step robot takes in the calculated direction from APF when attracting to the destination. d is a parameter that indicates a radius around the destination point; for faster position control, if the robot is outside radius d, more force will be applied to the robot to get closer to its destination faster.

By calculating the negative gradient of (6), we can achieve the attractive force that should be applied to the robot to reach its destination using

$$\vec{F}_{att,i} = \begin{cases} -\zeta\,\vec{r}_i & |\vec{r}_i| \le d \\ -d\,\zeta\,\frac{\vec{r}_i}{|\vec{r}_i|} & |\vec{r}_i| > d \end{cases} \tag{8}$$

After calculating the attractive force, we can define the repulsive potential field in (5), in which $U_{rep_{ij}}$ will be defined as

$$U_{rep_{ij}} = \begin{cases} \frac{1}{2}\,\eta\,(\frac{1}{|\vec{\rho}_{ij}|} - \frac{1}{\rho_o}) & |\vec{\rho}_{ij}| \le \rho_o \\ 0 & |\vec{\rho}_{ij}| > \rho_o \end{cases} \tag{9}$$

where the $\vec{\rho}_{ij}$ is a vector from the j^{th} obstacle to the i^{th} robot and is defined such as

$$\vec{\rho}_{ij} = \text{robot}\vec{[i]}.\text{pos} - \text{obstacle}\vec{[i]}.\text{pos}, \tag{10}$$

also η is a parameter that indicates the step robot takes in the calculated direction from APF when repulsing from obstacles. ρ_o is a parameter that indicates a radius around the obstacles; Unlike attractive force that affects the robot everywhere, the repulsive force only affects robots within a particular radius indicated by ρ_o.

By calculating the negative gradient of (9), we can achieve the repulsive force that should be applied to the robot to avoid obstacles using

$$\vec{F}_{rep,ij} = \begin{cases} \eta\,(\frac{1}{|\vec{\rho}_{ij}|} - \frac{1}{\rho_o})\,\frac{1}{|\vec{\rho}_{ij}|^2}\,\frac{\vec{\rho}_{ij}}{|\vec{\rho}_{ij}|} & |\vec{\rho}_{ij}| \le \rho_o \\ 0 & |\vec{\rho}_{ij}| > \rho_o \end{cases} \tag{11}$$

By summing the attractive force and all repulsive forces, the final robot's force will be calculated.

Remark. *Calculated forces from* (8) *and* (11) *and their sum are applied as the desired velocity (robot's input) to our single integrator model robots.*

A. ***Problem 1*** *Velocity and Inertia*

Assume we have highly dynamic robots with high velocity or robots with high inertia. In this case, the repulsive force in the APF method cannot dominate effectively, and the robot will not have sufficient time to avoid the obstacle. According to the repulsive force in the APF method, we have two parameters, η and ρ_o, to adjust the algorithm to suit our needs. Let us discuss how none of these parameters can solve this particular problem.

- η As discussed, η indicates our step length when an obstacle repels the robot. Increasing this parameter may seem to solve the issue, but in practical scenarios, the robots have limitations both on their maximum velocity and their maximum acceleration and deceleration; on this ground, increasing η only have a limited effect and will be saturated, eventually. Therefore it cannot solve the issue when the robot is going towards an obstacle with a high velocity.
- ρ_o This parameter defines a circle with ρ_o radius around the obstacle; the obstacle starts to repel a robot once the robot is within this circle. As a result, by increasing this parameter, we can solve our issue, temporarily. By increasing the radius around our obstacles, robots with high speed will detect obstacles sooner and have sufficient time to avoid them. However, in this scenario, robots with low velocities will be forced to avoid a much bigger circle

instead of the real obstacle. Thus this parameter is not the best option to deal with this problem too.

Solution. To solve this problem efficiently, we should take the robot's velocity vector into account. In order to do this, we make a prediction of the robot's future position using the current position and velocity using (12) and apply the repulsive force in relation to our prediction and not the actual position of our robot as it appeared in Fig. 9.

$$\vec{x}_{prediction} = \vec{x}_t + \vec{v}_t * \delta t \tag{12}$$

where the δt is the time we are predicting the position in the future.

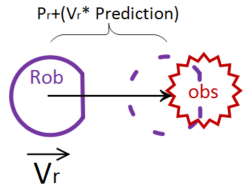


Fig. 9. Considering robot's prediction instead of its actual position when encountering an obstacle.

B. *Problem 2* Goal Non-Reachable with Obstacles Nearby (GNRON)

As the name suggests, this problem appears when the desired goal and destination of the robot is within an obstacle's virtual circle (defined by ρ_o). In this scenario, the repulsion force may paralyzes the robot to reach its goal.

Solution. to solve this problem, we add a term to our repulsive potential field [2] as

$$U_{rep_{ij}} = \begin{cases} \frac{1}{2}\,\eta\,(\frac{1}{|\vec{\rho}_{ij}|} - \frac{1}{\rho_o})\,|\vec{r}_i|^n & |\vec{\rho}_{ij}| \le \rho_o \\ 0 & |\vec{\rho}_{ij}| > \rho_o \end{cases} \tag{13}$$

in which, $|\vec{r}_i|^n$ is the added term to the equation, and the $\vec{r}_i$ is the same vector defined in (7), representing the distance between the robot and its destination, and n is the GNRON degree. Hence when the robot approaches its destination, the repulsive potential field decreases (higher the n, higher the amount of decrease) and allows the robot to reach its goal even if it is within an obstacle's radius.

By calculating the negative gradient of the (9), we achieve (14), which is the desired force applied to robots when repulsing from obstacles [2].

$$\begin{aligned} \vec{F}_{rep_{ij}} &= \begin{cases} \vec{F}_{rep_1} + \vec{F}_{rep_2} & |\vec{\rho}_{ij}| \le \rho_o \\ 0 & |\vec{\rho}_{ij}| > \rho_o \end{cases} \\ \vec{F}_{rep_1} &= \eta\,(\frac{1}{|\vec{\rho}_{ij}|} - \frac{1}{\rho_o})\,\frac{|\vec{r}_i|^n}{|\vec{\rho}_{ij}|^2}\,\hat{\vec{\rho}}_{ij} \\ \vec{F}_{rep_2} &= -\frac{n}{2}\,\eta * (\frac{1}{|\vec{\rho}_{ij}|} - \frac{1}{\rho_o})^2\,|\vec{r}_i|^{n-1}\,\hat{\vec{r}}_i \end{aligned} \tag{14}$$

where $\hat{\vec{\rho}}_{ij}$ and $\hat{\vec{r}}_i$ are the normalized vector of $\vec{\rho}_{ij}$ and $\vec{r}_i$, respectively. $(length = 1)$

C. *Problem 3* Local Minimums

The most common problem in APF is the problem of local minimums. Meaning a robot stucks in a local minimum of attraction and repulsion forces (where the attraction and repulsion forces neutralize each other) and never reaches its destination. The most common scenarios that lead a robot to be stuck in a local minimum are illustrated in Fig. 10.

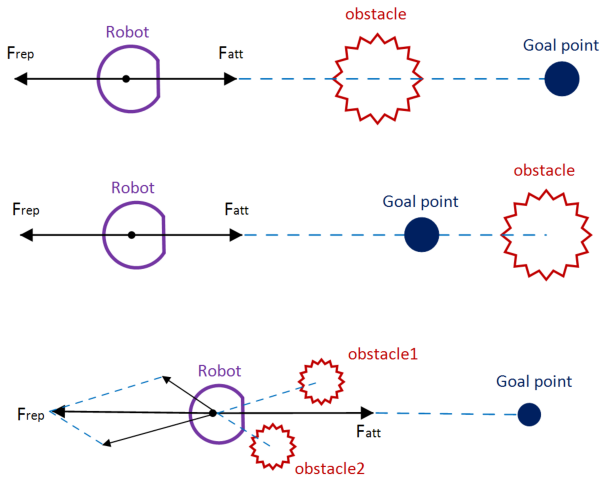


Fig. 10. Most common local minimums [2].

Solution. To solve this problem, we must first detect if a robot is stuck in a local minimum, and after the detection, we can loosen the robot by applying an additional force perpendicular to the attractive force, as shown in Fig. 11.

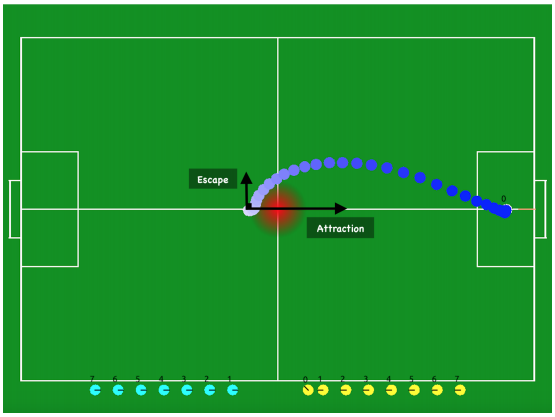


Fig. 11. Escape force when encountering a local minimum.

In order to detect the local minimum stuckness, we should detect if attractive and the sum of repulsive forces are neutralizing each other or not. However, these forces do not entirely neutralize themselves in practice, instead, we can use the Alg.2 to determine the local minimum.

Algorithm 2 Determining if a robot is in a local minimum

Input robot_position, destination, robot_velocity
Output local_minimum_decision

```
local_minimum_guesses = [ ]
while for several loops do
    local_minimum_guesses.append(false)
    if robot.velocity < threshold1 then
        #robot is stationary
        if dist(robot.position, destination) > threshold2 then
            #robot hasn't reached the destintion
            local_minimum_guesses[last] = true
        end if
    end if
end while
if all local_minimum_guesses is true then
    local_minimum_decision = true
else
    local_minimum_decision = false
end if
```

D. *Problem 4* Dynamic Obstacles

Another critical problem of the APF algorithm happens when the environment of the agents is dynamic, meaning the environment; specifically, the obstacles move over time. In this scenario, much like the first problem, the agents cannot detect the obstacles soon enough to avoid them ideally.

Solution. To give the robots sufficient time to detect and avoid the obstacles, predicting the trajectory of the robot and the obstacle can significantly mitigate the issue. To do this, we first consider the following line segments.

- rob_seg A segment that connects the robot's current position to the robot's position in δt second later. The latter can be calculated using (12).
- obs_seg A segment that connects the obstacle's current positions to the obstacle's position in δt second later. The latter can be calculated using (12).

Alg.3 shows our method of calculating the collision probability of our robots and a moving obstacle. The algorithm is further explained in the following.

Algorithm 3 Calculating the collision probability between a robot and an obstacle

Input $robot_{position}$, $obstacle_{position}$, $intersection$
Output collide_probability

```
if rob_seg 'has an intersection with' obs_seg then
    #There is a collision probability
    t_r = length(robot_position − intersection) / robot_velocity
    t_o = length(obstacle_position − intersection) / obstacle_velocity
    collide_probability = 1 / (1 + |t_r − t_o|)
end if
```

- **line 1.** Using robots and obstacle segments, we determine whether the robot's and obstacle's trajectories intersect. If there is an intersection, we proceed; otherwise, there is no collision probability that concerns us at least in the near future.
- **line 3.** We calculate the time takes for the robot to reach the intersection point at its current speed.
- **line 4.** We calculate the same thing as in line 3, but this time for the obstacle.

Fig. 12 shows a diagram of the segments and t_r and t_o

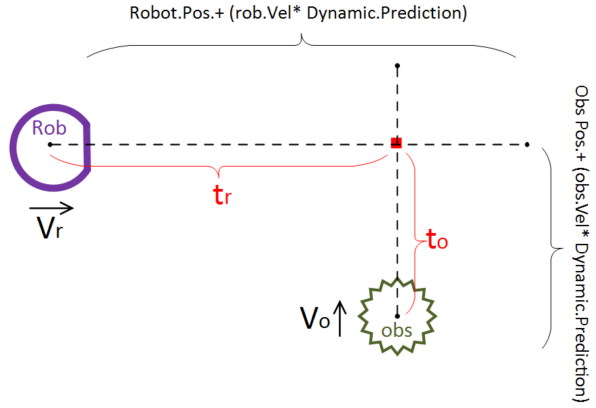


Fig. 12. collision probability scenario.

Using the obtained collision probability, we can now alter the repulsive force for each robot, using

$$\vec{F}_{rep_{ij}} = \vec{F}_{rep_{ij}}(\text{old}) + \alpha \times \textbf{collision\_probability} \times \hat{\vec{\gamma}}_j \quad (15)$$

in which, the $\vec{F}_{rep_{ij}}(\text{old})$ is the same repulsive force calculated in the GNRON section. α is a parameter that indicates the step robot takes to avoid the moving obstacle. $\hat{\vec{\gamma}}_j$ is a normalized vector that connects the intersection point to the robot's current position and is defined as

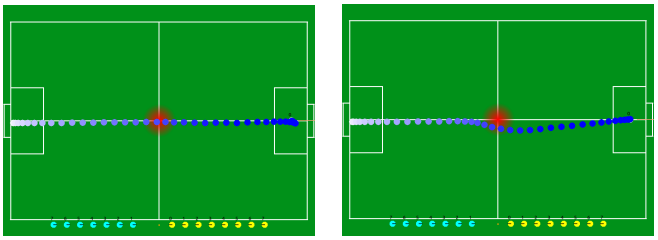
$$\hat{\vec{\gamma}}_j = \text{norm}(\vec{\text{obstacle}}[j].\text{pos} - \vec{\text{intersection point}}). \quad (16)$$

This vector ($\hat{\vec{\gamma}}_j$) indicates the escape direction of the robot, and it is defined as the agent would turn around the obstacle instead of moving forward.

Remark. *The $\vec{F}_{rep_{ij}}(old)$ or the GNRON solution of repulsion force will be calculated only for the obstacles that the robot is within their radius. On the other hand, the second term in* (15) *would be calculated for all the obstacles that intersect with the robot's trajectory.*

E. Experiments and Results

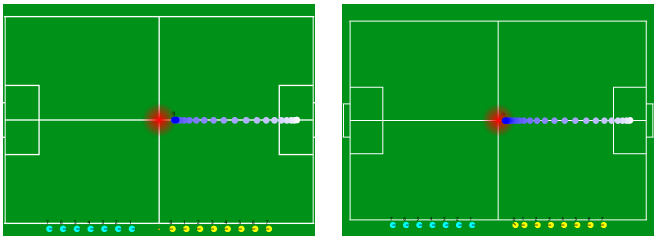
In Fig. 13, the trajectory of a robot moving with high velocity ($4\frac{m}{s}$) is shown in the presence of one obstacle. As shown in Fig. 13(a), the classic APF method cannot prevent the robot to avoid the obstacle at this speed. After applying the first proposed solution in Fig. 13(b), the robot will detect the obstacle at the right moment and avoid it altogether.



(a) Classic APF (b) Extended APF

Fig. 13. Solving the Velocity and Inertia problem.

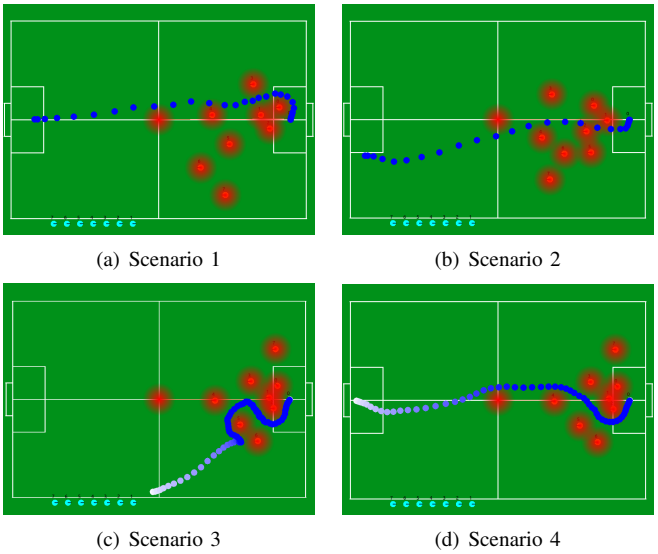
Fig. 14 illustrates the GNRON problem. In Fig. 14(a), the robot stucks in the obstacle's outer circle and cannot reach its destination. After applying the second proposed solution (GNRON solution), in Fig. 14(b), the robot reaches its destination within the obstacle's radius with no problem.



(a) Classic APF (b) Extended APF

Fig. 14. Solving the GNRON problem.

Fig. 15 shows examples of the EAPF method in 4 scenarios after applying all the solutions. The robot is moving with a maximum velocity of $4\frac{m}{s}$, and the field is $6 \times 9m^2$.

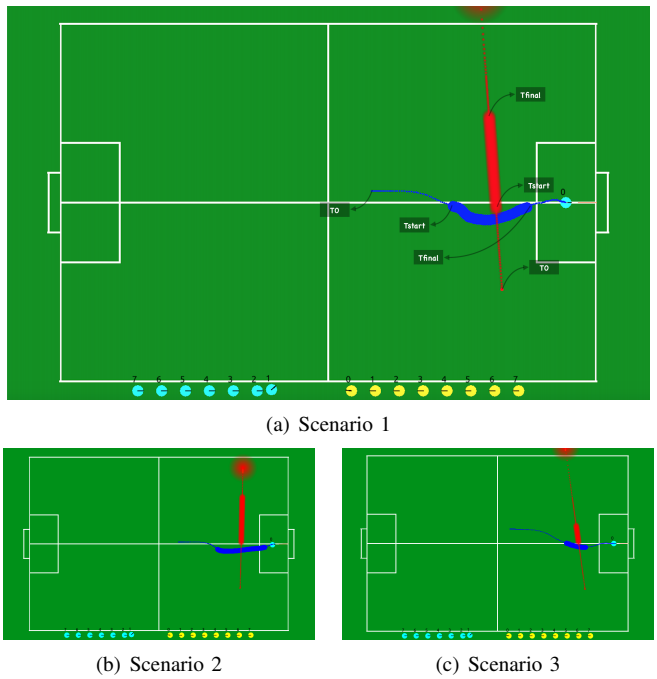


(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Fig. 15. Final EAPF result after applying all proposed methods.

Fig. 16 demonstrates three examples of the proposed EAPF when encountering with a moving obstacle. Both the obstacle and the robot start at T_0, and to have a better understanding of the timings and trajectories, the obstacle's and the robot's path have been bolded at the same time (T_{start}).



(a) Scenario 1

(b) Scenario 2 (c) Scenario 3

Fig. 16. Dynamic environment: robot vs. moving obstacle.

Conclusion

This paper presents modified Khatib's potential field algorithm applied to the soccer robot players. The proposed method can resolve the path planning problem of static and dynamic environments. Four known problems with Khatib's potential field algorithm (APF) has been introduced and addressed. The new EAPF algorithm evaluated under various simulations which carried out very feasible and efficient results demonstrating its effectiveness in both the static and the dynamic environments.

References

[1] G. X. Wang, G. H. Xu, G. Liu, W. J. Wang, and B. Li, "Fuzzy iterative sliding mode control applied for path following of an autonomous underwater vehicle with large inertia," Mathe- matical Problems in Engineering, vol. 2019, Article ID 8650243, 14 pages, 2019.

[2] X. Fan, Y. Guo, and B. Wei, "Improved Artificial Potential Field Method Applied for AUV Path Planning," Mathematical Problems in Engineering, March 2020.

[3] S. M. LaValle, "Planning Algorithms," Cambridge University Press, New York,NY, USA, 2006.

[4] J. Han and Y. Seo, "Mobile robot path planning with surrounding point set and path improvement," Applied Soft Computing, vol. 57, pp. 35–47, 2017.

[5] Y. Wang, D. Mulvaney, I. Sillitoe, and E. Swere, "Robot navigation by waypoints," Journal of Intelligent and Robotic Systems, vol. 52, no. 2, pp. 175–207, 2008.

[6] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning a chronological review," Proceedings of World Academy of Science: Engineering & Technology, vol. 23, pp. 101–106, 2007.

[7] P. Bhattacharya and M. Gavrilova, "Roadmap-based path planning–using the voronoi diagram for a clearance-based shortest path," IEEE Robotics & Automation Magazine, vol. 15, no. 2, pp. 58–66, 2008.

[8] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," IEEE Trans- actions on Systems Man and Cybernetics Part B-Cybernetics, vol. 39, no. 3, pp. 672–689, 2009.
[9] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.
[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol. 1, no. 1, pp. 269–271, 1959.
[11] M. Kurant, A. Markopoulou, and P. Thiran, "Towards unbiased BFS sampling," IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1799–1809, 2011.
[12] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, "Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation," IIEEE Transactions on In- dustrial Electronics, vol. 58, no. 10, pp. 4813–4821, 2011.
[13] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," The International Journal of Robotics Research, vol. 5, no. 1, pp. 90–98, 1986.
[14] V. Ghidary, "grSim – RoboCup Small Size Robot Soccer Simulator," Springer, Berlin, Heidelberg, vol. 7416, 2012.
[15] J. Biswas, "ROBOCUP SMALL SIZE LEAGUE," Robocup Competitions, [Online]. Available: ssl.robocup.org.
[16] S. Zickler, T. Laue, and O. Birbach, "SSL-Vision: The Shared Vision System for the RoboCup Small Size League," Springer, Berlin, Heidelberg, vol. 5949, 2010.
[17] A. Maranhão, A. Schuch, A. Azevedo, "ITAndroids Small Size League Team Description Paper for RoboCup 2020,".
[18] C. Li, T. Watanabe, Z. Wu, "The Real-Time and Embedded Soccer Robot Control System," Vladan Papić, IntechOpen, DOI: 10.5772/7352. Available from: https://www.intechopen.com/chapters/9346.