Array 14 (2022) 100134

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

Computational Time Reduction in Stream Field Navigation Through Dynamic Path Search and Selection in Kinematic Environments

Hussein M. Fawzy *, Hisham M. El-Sherif, Gerd Baumann

German University in Cairo, Cairo, Egypt

ARTICLE INFO

Keywords:
Path planning
Mobile robot
Navier-Stokes
Path algorithm
Fluid dynamics
Kinematic environments

ABSTRACT

Path planning algorithms based on physical models have been in development to allow navigating an environment in a way that mimics nature. Contrary to combinatorial and sampling-based algorithms, which represent the field by a set of nodes then use searching techniques to find a path, the artificial potential field method solves this problem by attracting the robot to the target and repelling it from the obstacles While the Stream Field Navigation (SFN) algorithm solves the problem by simulating a fluid field. This research proposes a dynamic path search and selection schema to allow the SFN algorithm to deal with moving obstacles. This research shows how the algorithm deals with a kinematic (variable) environment using the discretized path calculations. The dynamic search and selection algorithm is also capable of updating the robot's path when the goal is re-located. The developed approach is shown to greatly reduce the computational time required to obtain a path compared to re-simulating the field when a change is introduced to the environment.

1. Introduction

The computational problem of finding a collision-free path between two points, A and B, based on given constraints is classically called the path planning problem and solving it is considered as a fundamental functionality for autonomous and semi-autonomous systems.

The path planning problem is solved in a fully defined environment. That is, the field boundaries, configuration and obstacles are defined. These definitions are obtained prior to the path planning phase by, for example, Simultaneous Localization and Mapping algorithms [1–3].

The path planning problem has been previously approached by multiple classes of algorithms; such as Combinatorial Planning [4,5], Sampling-Based Planning techniques [6,7] and physical algorithms which include the Artificial Potential Field method (APF) [8–10] and the approach of interest in this paper, the Stream Field Navigation approach [11].

Each of these classes has benefits and limitations. For example, algorithms in the combinatorial planning class are complete (A path will be found if it exists), however, they become quickly inefficient as the dimensionality of the field space increases. Sampling-based techniques are probabilistically complete (the probability of finding this solution approaches one if the algorithm kept working without bounds) while in the artificial potential field, there is the problem of local-minima being created in the field.

The local minima problem in the APF algorithms can be solved by estimating obstacles to be cylinders and then calculating fluid stream only around these cylinders using specific potential functions [15]. The downside in this solution is that complex obstacles are approximated to be cylinders and it may not be a valid approximation in some cases. Another solution is also introduced in [16]. Machine learning techniques [12] and optimization techniques [13,14] were also applied to improve the obstacle avoidance behavior of the artificial potential field.

The Stream Field Navigation algorithm uses fluid stream equations and mechanics to represent the navigation field and inherently solves the local-minima problem. Although the SFN algorithm is improved the computational efficiency as it uses the directional residuals, it would still require the stream directions to be recalculated in a kinematic environment when the obstacles change their position.

This research introduces another layer of efficiency improvement for the case of the mentioned kinematic environments. In the next section, the mathematical model and the fundamental SFN approach are explained while section 3 shows how a path can be dynamically updated to avoid stream re-calculations in special cases. Section 4 show the results and time improvement of the introduced approach.

* Corresponding author.
E-mail addresses: hussein.fawzy.1@gmail.com (H.M. Fawzy), hisham.elsherif@guc.edu.eg (H.M. El-Sherif), gerd.baumann@guc.edu.eg (G. Baumann).

https://doi.org/10.1016/j.array.2022.100134
Received 2 October 2021; Accepted 22 February 2022
Available online 16 March 2022

2. The mathematical model and the fundamental SFN approach

The SFN algorithm is based on the Navier-Stokes equations to allow determining the velocity vector field for the navigation problem. The equation for the x direction of a 3D incompressible flow is:

$$\rho a_x = \rho g_x - \frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

This equation can be generalized for the y and z directions as well and it is composed of the gravity force, pressure force and viscosity force. There are two important points to note in this mathematical model. One, there exists a continuous solution for these equations on $\mathbb{R}^3 x[0,T]$ [17]. (The is beyond the scope of this research. For the SFN algorithm, it guarantees that a solution exists for a given field given that the domain of the time dimension is $[0,T]$ where $T < \infty$). And two, numerically solving the Navier-Stokes equations yields the velocity vector field for a given domain; I.E., a vector field for each point in the discretized domain. These two points together form the basis for the dynamic path search and selection introduced in this paper.

A solution to a general path planning problem is obtained using the SFN algorithm by following set of steps that were developed to be systematic (Fig. 1).

The field parameters (ex. kinematic viscosity and inlet velocity components) and the field and obstacle geometry are defined first. Next, a meshing algorithm is used to produce a mesh of the field, as seen in Fig. 2, where the Navier-Stokes system of equation will be numerically solved.

The generated mesh is critical to how the dynamic path update selects a new path to allow reaching the goal point as explained in Section 3. That is because the mesh points are where the field quantities (velocities and pressure) are calculated.

The SFN algorithm then defines the initial and boundary conditions of the navigation problem before numerically solving the stream equations.

At this step, a path can simply be constructed from one discrete velocity vector $\boldsymbol{v_n}$ to the next velocity vector $\boldsymbol{v_{n+1}}$ by following the stream directions starting at $\boldsymbol{q_s}$ until a complete path is formed between $\boldsymbol{q_s}$ and $\boldsymbol{q_g}$ using N velocity vectors ranging from $\boldsymbol{v_0}$ to $\boldsymbol{v_N}$ (Fig. 3).

The Reynolds number $Re = \frac{uL}{\nu}$ is chosen to keep the flow laminar, characterized by a smooth and constant fluid motion, as any turbulent effect will increase the problem difficulty for robotic applications. In the equation, $\boldsymbol{u}$ is the velocity of the stream vector, $\boldsymbol{\nu}$ is the kinematic viscosity of the fluid and $\boldsymbol{L}$ is a characteristic linear dimension of the field.

The critical Reynolds number for a turbulent flow in a range from 10^5 to 10^6 [24]. In the SFN algorithm, the Reynolds number is set to a lower value.

The important note here is that the SFN has the inherent ability to always generate a path from the start point $\boldsymbol{q_s}$ to the goal point $\boldsymbol{q_g}$ as long as a path exists. This property is guaranteed as the underlaying physical model ensures the conservation of momentum and mass. In other words, it guarantees that if there is mass flowing inside the system, there will be mass flowing outside.

3. Navigation in kinematic fields

Kinematic (or variable) environments are environments where the field is not static. In these environments, it is more feasible to think of the obstacles as humans or even other robots. Such field obstacles are usually moving around the environment to complete tasks of their own. The target (goal) is also allowed to move in a variable environment (For example, a robot may be following another robot). The variable environment problem is unique in nature because it is a problem faced by the robot after the path is calculated and the robot is set to follow it. This would require the algorithm responsible for the path planning to actively update the path to ensure reaching the same target with no collisions.

A moving obstacle inside the navigation field is not necessarily going to collide with the robot. There are three cases for a moving obstacle and its interaction with the calculated path of the robot:

1. The observed path of the moving obstacle does not intersect with the calculated path of the robot.
2. The observed path of the moving obstacle intersects with the calculated path of the robot. However, the obstacle and the robot do not reach a certain position $\boldsymbol{p}$ at the same time $\boldsymbol{t}$.
3. The observed path of the moving obstacle intersects with the calculated path of the robot and they reach a certain position $\boldsymbol{p}$ at the same time $\boldsymbol{t}$.

To deal with the variable environments problem, an environment mapping algorithm is required to first calculate the trajectories of the kinematic obstacles in the environment and then decide which of the above three cases to follow.

Assuming that the position of the robot at time $\boldsymbol{t}$ is $\boldsymbol{p_r}(t)$ where:

$$p_r(t) = x_r(t)\cdot i + y_r(t)\cdot j$$

And the position of the kinematic obstacle at time $\boldsymbol{t}$ is $\boldsymbol{p_k}(t)$ where:

$$p_k(t) = x_k(t)\cdot i + y_k(t)\cdot j$$

If there exists a real and positive time $\boldsymbol{t_{collision}}$ that satisfies the following two equations, then a collision is going to happen between the robot and the kinematic obstacle unless the robot updates its path. In which case, the environment mapping algorithms reports that the current case is case three to the path planning algorithm.

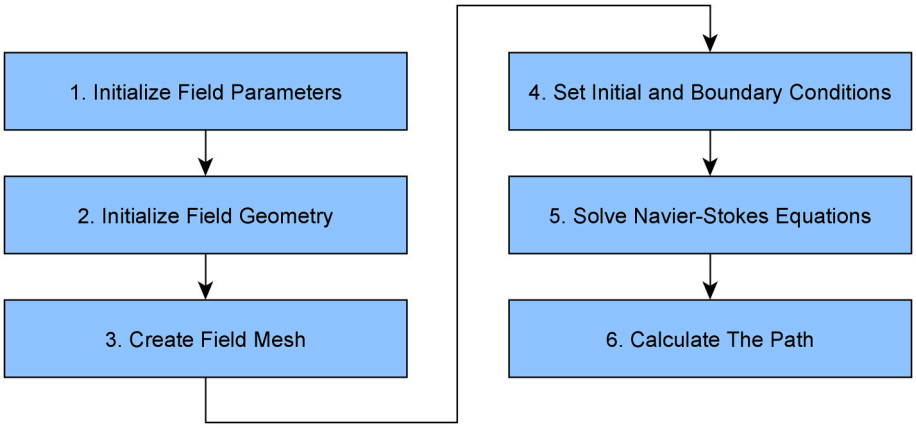

Fig. 1. Stream Field Navigation Algorithm Steps [11].

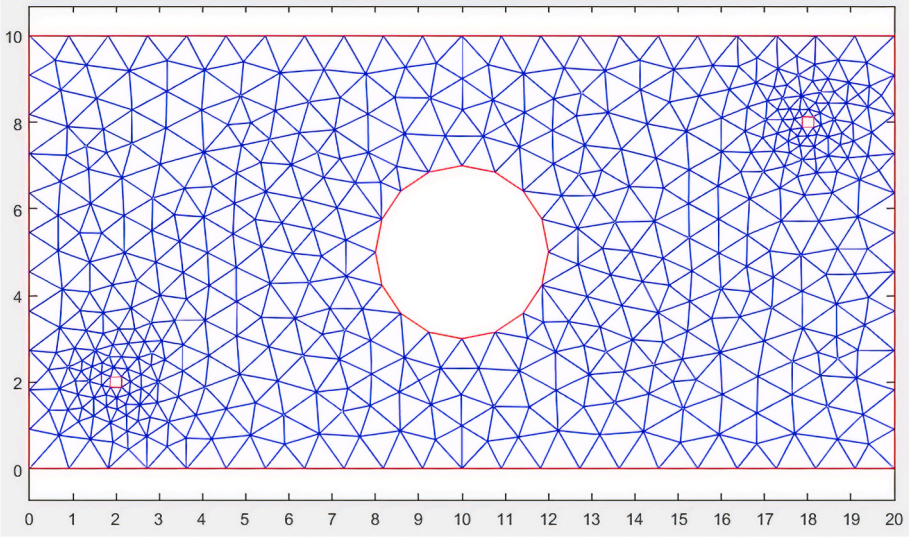

Fig. 2. Mesh of the simple navigation field [11].

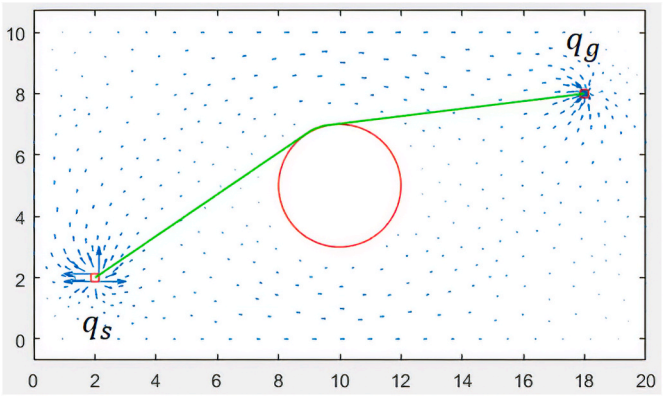

Fig. 3. Path Between Start and goal points [11].

$$x_r(t_{collision}) = x_k(t_{collision})$$

$$y_r(t_{collision}) = y_k(t_{collision})$$

If $t_{collision}$ is not real or negative, then its either *case one* or *case two* that will be reported by the environment mapping algorithm and the path planning algorithm does not need to take any actions to update the path of the robot. Because of this, the upcoming part of this section deals with case three of kinematic obstacle observations.

Due to the multiple encounters of variable environments in the field of robotics, researchers are always searching for new method to deal with these environments [28]. Some of the methods that are researched for dealing with this problem are using optimization techniques [29] and a combination of exhaustive and heuristic search techniques [30].

There are generally two approaches when dealing with the variable field problem, the online approach (The local approach) or the offline approach (The global approach). A robotic system using the former approach updates the path while navigating the environment and while the robot is moving to its target. As for the latter approach, the path is re-calculated from the beginning before the robot takes any more actions.

In case of an upcoming collision, the local field planner processes the intersection between the robot's path and field obstacles then manipulates the path to avoid the collision.

First, the velocity of the robot can be updated so that the robot and the obstacle do not pass through the collision point at the same time. In other words, the Local Field Planner can change the velocity of the robot to either move faster or slower to avoid the collision. To mathematically model this case, the position equations are first generalized as follows:

$$p_r(t, v_r) = x_r(t, v_{rx}) \cdot i + y_r(t, v_{ry}) \cdot j$$

$$p_k(t, v_d) = x_k(t, v_{dx}) \cdot i + y_k(t, v_{dy}) \cdot j$$

where:

$$v = \sqrt{v_x^2 + v_y^2}$$

The conditions for an upcoming collision are then modeled as:

$$x_r(t_{collision}, v_{rx}) = x_k(t_{collision}, v_{dx})$$

$$y_r(t_{collision}, v_{ry}) = y_k(t_{collision}, v_{dy})$$

To avoid the collision, the Local Field Planner would change v_r to v_r' within a certain range of possible robot velocities and without changing the direction of robot motion

$$v_r' \in [v_{r-min} : v_{r-max}]$$

If the updated velocity satisfies any of the following equations, then the collision is avoided

$$x_r(t_{collision}, v_{rx}') \neq x_k(t_{collision}, v_{dx})$$

$$y_r(t_{collision}, v_{ry}') \neq y_k(t_{collision}, v_{dy})$$

where:

$$v' = \sqrt{v_x'^2 + v_y'^2}$$

In some cases, it is not possible to find a velocity v_r' that avoid a collision and there are two reasons for this, one, the velocity that would avoid the collision falls outside the possible range of robot velocities $v_r' \notin [v_{r-min} : v_{r-max}]$ and so it is not a possible solution, and two, the kinematic obstacle has stopped its motion while it is on the path of the robot so there is no possible velocity change without changing the direction of the robot that would avoid the collision. An example for this is shown in Fig. 4 where the obstacle in the field changed its position and stopped to intersect the robot's path. The figure also shows the multiple paths extracted to connect q_s to q_g; which was possible due to the fact that the velocity components are known at each mesh point.

The dynamic path selection algorithm can solve this problem using

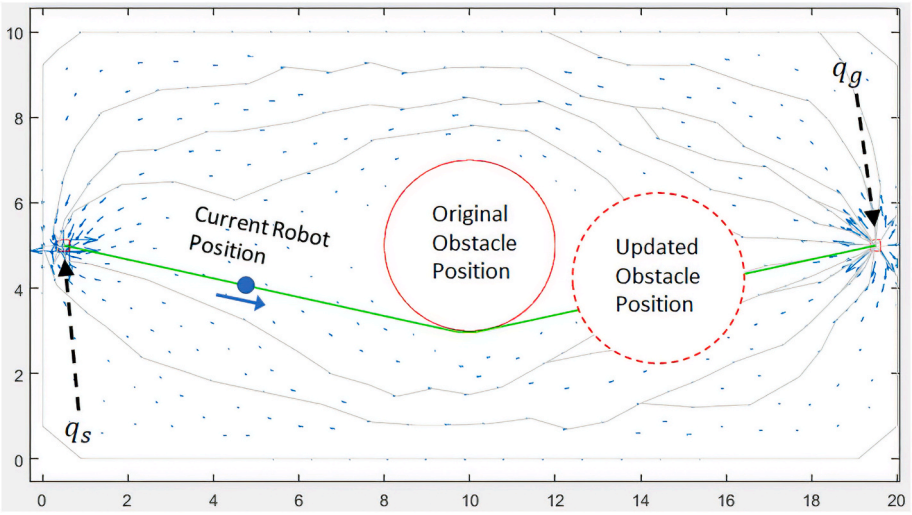

Fig. 4. Updated obstacle position in a variable field.

the fact that the velocity components are known at each mesh point and that multiple paths have already been extracted to connect $\boldsymbol{q_s}$ to $\boldsymbol{q_g}$.

Let $\boldsymbol{U}$ be the set of all calculated streams (paths) in a given field, $\boldsymbol{U_g}$ be the sub-set of streams that successfully reach from $\boldsymbol{q_s}$ to $\boldsymbol{q_g}$ and $\boldsymbol{U_0}$ be the sub-set of streams that do not reach from start to goal or get trapped in a stagnation point.

$$U = U_g \cup U_0$$

To navigate around a re-located obstacle in Fig. 4 the algorithm first splits the calculated paths into two sets, a set that is now not suitable for the robot as an obstacle has blocked it ($\boldsymbol{P_{blocked}}$) and the other set is clear and the robot can use it to reach its goal ($\boldsymbol{P_{clear}}$) so that:

$$U_g = P_{blocked} \cup P_{clear}$$

The sub-set $\boldsymbol{P_{blocked}}$ is then removed from the set possible robot paths $\boldsymbol{U_g}$, the algorithm now selects the most optimum path from the $\boldsymbol{P_{clear}}$ sub-set; in this case, the shortest path.

As any other calculated path in the field, the selected optimum path is a series of velocity vectors ranging from $\boldsymbol{v_0}$ to $\boldsymbol{v_N}$ where $\boldsymbol{N}$ is the number of velocity vectors on the path. The algorithm loops over the velocity vectors of the selected optimum path using a variable $\boldsymbol{i}$ ranging from 0 to … At each velocity vector $\boldsymbol{v_i}$, the algorithm tries to connect the current robot position to $\boldsymbol{v_i}$ using a straight line and if this straight line does not intersect any of the field obstacle, then a new path has been formed and the collision has been avoided. However, if the new straight line intersects with an obstacle, then the local field planner applies the same a routine to $\boldsymbol{v_{i+1}}$ until a collision-free straight line is found. The result of applying the collision avoidance routine is seen in Fig. 5.

The previous method to avoid a collision falls inside the online updates category where the path update is executed locally on the robot and no major changes has been made to the simulated environment. It greatly reduces the computation time as it prevents the re-calculation of velocity vector every time an obstacle is re-located.

However, there are cases where the online updates will not solve the collision-free path planning problem which would require the field velocities to be re-calculated for the updated field.

Let $\boldsymbol{F_{Space}}$ define the free space of the field where the robot is allowed move and $\boldsymbol{O_{Space}}$ define the obstacle space, i.e., the space taken by the obstacles and the robot is not allowed to intersect with. By definition:

$$F_{Space} \cap O_{Space} = \varnothing$$

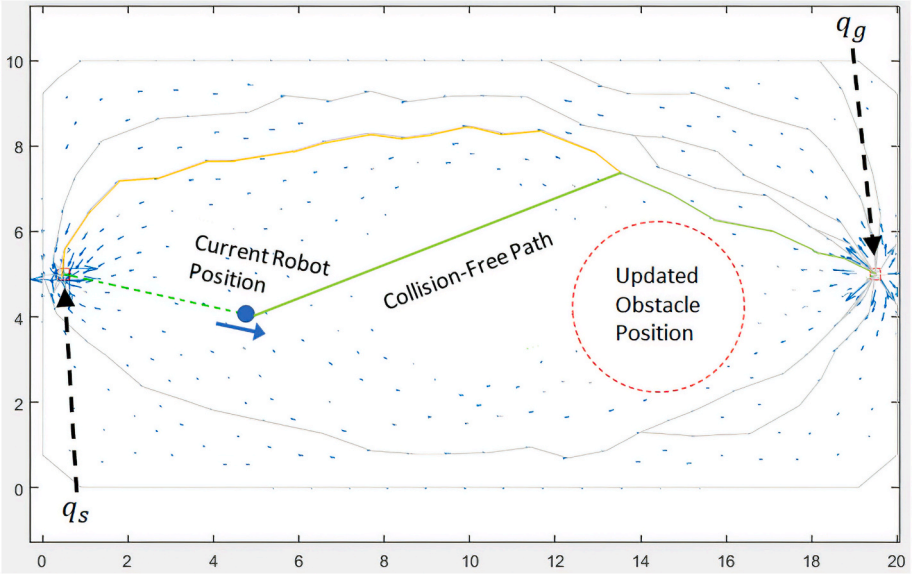

Fig. 5. Variable field collision-free path.

There exist one or more sets of mesh points, M_{cj}, that when blocked, the possibility of connecting q_s and q_g using only F_{Space} no longer exists. These sets of mesh points are referred to as the *critical sets*, M_c.

$$M_{cj} \in M_c$$

$$M_{cj} = \{v_i : v_i \in F_{Space} \text{ and } 0 \le i \le N_j\}$$

where N_j is the number of points in critical point set j and v_i represents a velocity vector or a mesh point in that set (since the two are defined for the same locations in the mesh).

Two sets of critical points are shown in Fig. 6, $\{M_{c1}, M_{c2}\} \subset M_c$. If M_{c1} is blocked by any number or any shape of obstacles, there would be no path to connect q_s to the q_g as the field to be split into two fields where q_s falls inside one field and q_g falls inside the other.

The second set of critical points, M_{c2}, is different in nature from M_{c1}. The fact that if the two field obstacles were re-located to block M_{c2}, the field would not be split into two parts and it would still be possible to connect q_s to q_g. However, the free space that would be created between the two obstacles was not discretized and therefore, has not stream velocities calculated. In this case, the SFN algorithm has to carry out an offline path update and recalculate the stream velocities and pressures so that a new path can be formed that connects the required points together.

As mentioned earlier, in a variable environment, the goal is also allowed to be re-located. The kinematic goal environment can be explained by the field in Fig. 7. The field has two obstacles, the streams of the field are calculated and the robot's path is constructed. While the robot is moving on its path, one of the obstacles is moved to obstruct the path and the goal is moved as well to another location.

This approach used to account for moving obstacles is slightly updated to also account for the moving goal. The idea is to use the already calculated set of streams that cover the field to connect the agent to the updated goal location. If there exists a stream that both the agent and the – re-located – goal can be connected to using linear and collision-free transitions (Straight line transitions), then this stream can be used to construct a path that connect the current agent location to the updated goal location.

The algorithm first constructs the sub-set of clear paths, P_{clear}, by removing the paths that are intersecting any of the obstacles from the set U_g. Similar to the approach followed when only the obstacles are kinematic, the algorithm then selects the optimum path from the sub-set P_{clear} as seen in Fig. 8.

Using the selected optimum path, consisting of N velocity vectors ranging from v_0 to v_{N-1}, the task at this point is to connect the goal point q_g to a point v_i on the path that minimizes the function $d(q_g, v_i)$ where:

$$d(q_g, v_i) = \sqrt{(v_{i,y} - q_{g,y})^2 + (v_{i,x} - q_{g,x})^2}$$

If the algorithm cannot find a collision-free and linear transition between q_g and any of the vectors on the currently selected path, the next optimum path is chosen and the same steps are followed again until a transition line between q_g and the path is found.

The point on the selected optimum path that the goal q_g can be transitioned to is labeled v_g (Velocity vector g). The algorithm now tries to connect the current agent location q_r to the same path by looping over the velocity vectors of the selected optimum path using a variable i ranging from g to 0 and from g to $N-1$. The target is to transition the agent from its current location q_r to a location v_r on the selected optimum path closest to the goal transition point v_g as can be seen in Fig. 8. Once this transition is found, an updated path can be constructed as follows:

$$q_r \rightarrow v_r \rightarrow v_g \rightarrow q_g$$

4. Verification and results

The dynamic path search and selection algorithm was tested against the original SFN algorithm to show how the updated paths compare to the original paths and to show the difference in execution times between re-running the SFN algorithm and using the dynamic path search and selection algorithm when obstacles are moved inside the field. The test is carried out in two different fields.

The first test field, labeled Test Field A (Shown in Fig. 9), includes four similar obstacles where the right most obstacle is re-located while the agent is moving on the path.

For the second test field, Test Field B (Shown in Fig. 10), the field includes a set of randomly generated convex and concave obstacles to test the behavior of the dynamic path search and selection algorithm in cluttered environments. In this field, two of the obstacles and the goal are re-located after the original path calculation by the SFN algorithm and before the agent starts moving through the environment.

The original and updated obstacle configurations as well as the original and updated paths are shown in the figures for path comparisons. In these figures, the original path is shown in gray, the updated path is shown in green, the original obstacles are represented with a dashed black outline and the final obstacle configurations are represented with a red outline.

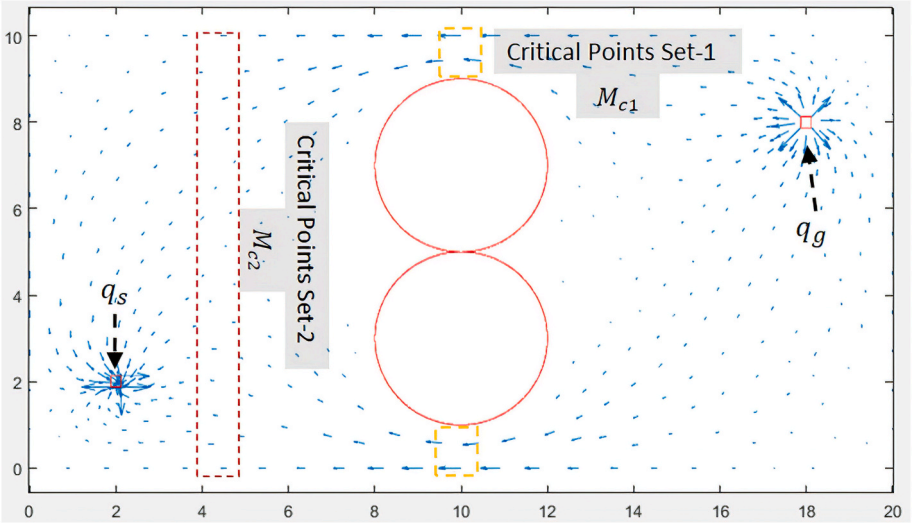

Fig. 6. Examples of critical point sets.

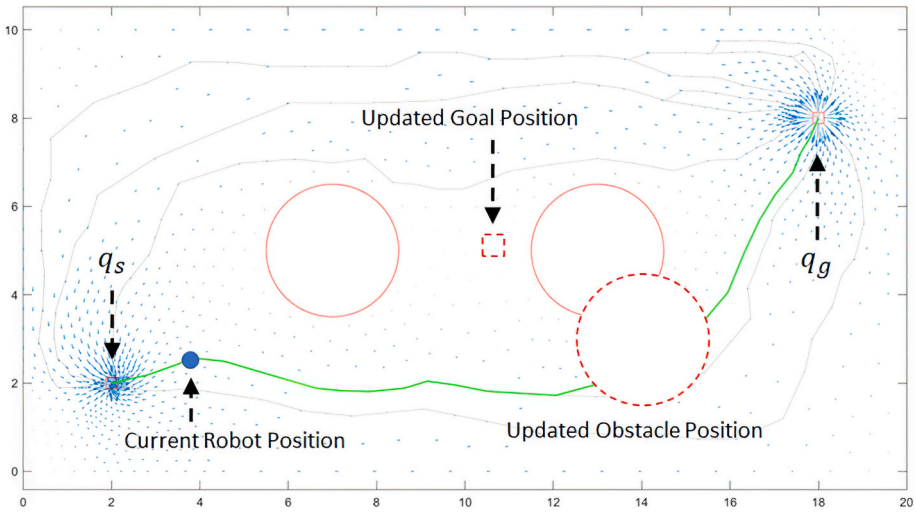

Fig. 7. Kinematic goal and obstacles field.

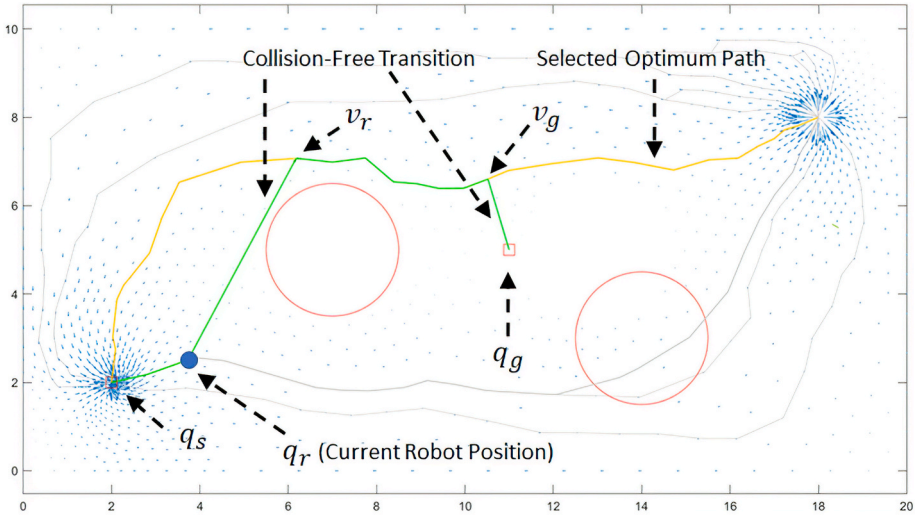

Fig. 8. Kinematic goal and obstacles field - updated path.

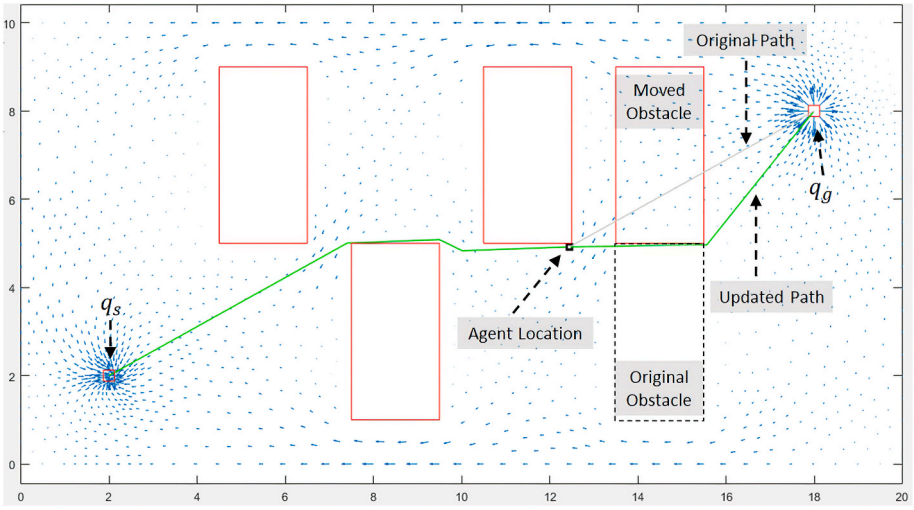

Fig. 9. Test field A

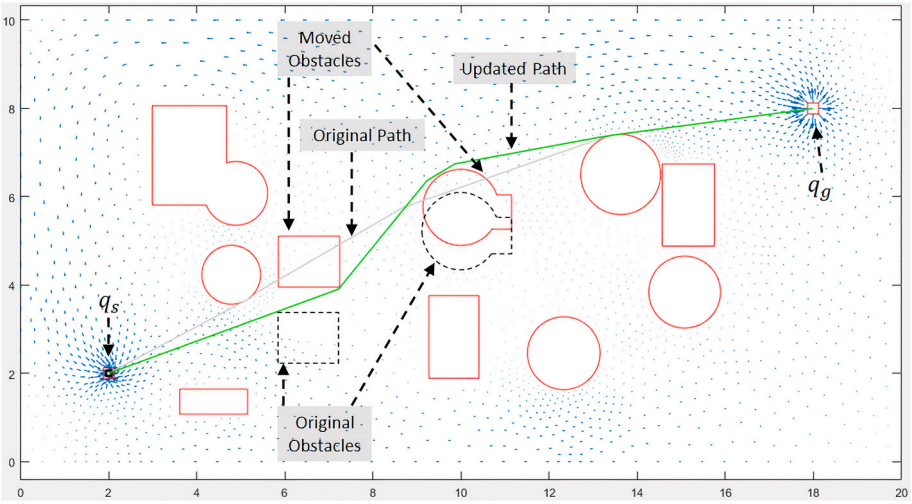

Fig. 10. Test field B.

The experiment is executed 100 times in each field to obtain empirical statistical results of execution times. For each field, the results are expressed as the mean (average) execution time and the standard deviation of the execution times of the 100 runs. The standard deviation is included in the results as the mean by itself is not an effective statistical measure for execution times [31].

The results for Field A and Field B are shown in Table 1 and Table 2 respectively. All the tests were executed on an Intel 6500U Mobile Processor with 8 GB of RAM.

The results show that there is an improvement, in terms of execution times, when dynamic path search and selection is used. This is mainly because the mesh generation for a field and numerical solution process of the Navier-Stokes equations are major contributors in the execution times of the SFN algorithm (While obtaining a path for fields A and B using the SFN algorithm, the mesh generation the numerical solution process accounted for approximately 74% of the execution times, on average). Using the dynamic path search and selection algorithm eliminates the need for rebuilding the mesh redoing the numerical calculations.

In Test Field A, there is approximately a 92% improvement in execution times between the two re-running the SFN algorithm and dynamically updating the path while for Test Field B, the improvement is 78%. The improvement is lower for Test Field B due to the higher number of obstacles and the complex nature of the paths generated between these obstacles which increases the time required to construct an updated path.

Table 1
Field A results.

Algorithm	SFN	Dynamic Path Search and Selection
Mean (s)	0.241	0.020
Standard Deviation (s)	0.004	0.002

Table 2
Field B results.

Algorithm	SFN	Dynamic Path Search and Selection
Mean (s)	0.577	0.131
Standard Deviation (s)	0.006	0.009

5. Conclusion

The developed SFN algorithm has been proven to find a collision-free and local-minima-free path in a navigation field. However, since the SFN algorithm is based on the Navier-Stokes equations, it is important that the execution time of using the algorithm is minimized.

The target of this research work was to develop a method for the SFN algorithm so that it can be used in kinematic environments without incurring the major computational cost or numerically solving the Navier-Stokes equations for kinematic fields. This target is realized using the developed dynamic search and selection algorithm. The algorithm is shown to successfully find a path around re-located obstacles in different environment configurations with convex and concave obstacles without recalculating the velocity vectors using the SFN algorithm. It was also shown that the dynamic search and selection algorithm is able to reconstruct a path from the agent to the goal when the goal is re-located. The execution time comparisons show that the dynamic search and selection algorithm has a substantial improvement to the application efficiency compared to re-running the SFN algorithm when obstacles are re-located.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Csorba M, Uhlmann J. A suboptimal algorithm for automatic map building. American Control Conference; 1997.
[2] Jaulin L. A nonlinear set-membership approach for the localization and map building of an underwater robot using interval constraint propagation. IEEE Trans Robotic. 2009;25:88–98.
[3] Jaulin L. Range-only SLAM with occupancy maps; A set-membership approach. IEEE Trans Robotic. 2011;27(5):1004–10.
[4] Streinu I. A combinatorial approach to planar non-colliding robot arm motion planning. In: Foundations of computer science. CA, USA: IEEE; 2020.
[5] Nguyet TTN, Hoai TV, Thi NA. Some advanced techniques in reducing time for path planning based on visibility graph. In: International conference on knowledge and systems engineering. Hanoi: IEEE; 2011.
[6] Morgan S, Branicky M. Sampling-based planning for discrete spaces. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), sendai, Japan; 2004.
[7] Kingston Z, Mol M, Kavraki LE. Sampling-based methods for motion planning with constraints. Ann Rev Contr Robot Autonom Syst 2018;1:159–85.
[8] Khatib O. Real-time 0bstacle avoidance for manipulators and mobile robots. In: Ieee int. Conf. On robotics and automation; 1985.
[9] Chen Y-b, Luo G-c, Mei Y-s, Yu J-q, Su X-l. UAV path planning using artificial potential field method updated by optimal control theory. Int J Syst Sci 2016;47 (6).
[10] Wu Z, Hu G, Feng L, Wu J, Liu S. Collision avoidance for mobile robots based on artificial potential field and obstacle envelope modelling. Assemb Autom 2016;36 (3).

[11] Fawzy HM, El-Sherif HM, Baumann G. A framework for robotic path planning based on enhanced fluid potential dynamical models. In: International conference on Mechatronics and robotics engineering, munich, Germany; 2022.
[12] Michels J, Saxena A, Ng AY. High speed obstacle avoidance using monocular vision and reinforcement learning. In: International conference on machine learning. Bonn, Germany: ICML; 2005.
[13] Palm R, Bouguerra A. Navigation of mobile robots by potential field methods and market-based optimization. In: European conference on mobile robots. Sweden: Oerebro; 2011.
[14] Palm R, Bouguerra A. Particle swarm optimization of potential fields for obstacle avoidance. RARM; 2013.
[15] Palm R, Driankov D. Fluid mechanics for path planning and obstacle avoidance of mobile robots. In: AASS. Orebro University; 2014.
[16] Chang H. A new technique to handle local minimum for imperfect potential field based motion planning. In: Ieee int. Conf. On robotics and automation; 1996.
[17] Stenger F, Tucker D, Baumann G. Navier–Stokes equations on r3x[0, T]. Springer; 2016.
[24] White FM. Fluid mechanics. McGraw-Hill; 2002.
[28] Gupta NK, Vishwakarma SK, Pandey P, Nesaraj SO. Robot navigation in dynamic environment. Int J Innov Eng Technol 2016;7(4).
[29] Hossain A, Ferdous I. Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. Robot Autonom Syst 2015;64:137–41.
[30] P. Fiorini and Z. Shiller, "Robot motion planning in dynamic environments," in Robotics research.
[31] Munoz P, Barrero DF, R-Moreno MD. Run-time analysis of classical path-planning algorithms. Res Develop Intell Syst 2012:137–48.

Prof. Dr. Eng. Hisham El-Sherif. He received my B.Sc. degree with “Distinction with Honor’s Degree” in Electronics and Communications Engineering from Ain Shams University, Egypt, the M.Sc. degree in Electrical Engineering from Ain Shams University, Egypt, and the Ph.D. degree in Neural networks for speech recognition Engineering from Ain Shams University, Egypt in 1998. During his academic study years, He worked with Siemens, Department of Systems Engineering and Automation. From 2006 to present; Dr. Hisham is working at the German University in Cairo as a head of Industrial Automation Department. He is also director of Robotics and Autonomous Systems (RAS) research group at German University in Cairo. He worked as a Senior Research Scientist at Vestec, Inc and AI Consultant at Menya Solutions Inc., a Robotics & Automation Consultant at InnoVision Systems, and Head of Robotics at Sypron Solutions. Dr. Hisham is also a Member in the IEEE, and chair of IEEE Robotics and Automation Society (RAS) Egypt Chapter (recipient of the 2015 IEEE RAS Chapter of the Year Award and 2012 Chapter of the Year Award in IEEE Region 8). Reviewer and Session Chair of 2020 6th International Conference on Mechatronics and Robotics Engineering.

Prof. Dr. Gerd Baumann. Gerd Baumann received the Dipl.-Phys. degree (Hons.) in 1985 and the Ph.D. degree (Hons.) in mathematical physics in 1988 from the University of Ulm, Ulm, Germany, respectively. In 1993, he received the Venia Legendi for theoretical physics and was first appointed as a professor in 1998 with the University of Ulm. In 1989, he was a Researcher with the Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM, USA. Besides his university activities, in 2000, he was the Head of advanced engineering and consultant with major international companies in automotive, medicine, software engineering, and biology. In 2004, he was a Professor and the Head of the Mathematics Department with German University in Cairo, Cairo, Egypt. He managed different bilateral international research cooperations funded by BMBF between Germany and Egypt, in the field of lightweight and degradable materials in biological applications and lightning of ancient buildings. His research interests include applied hybrid mathematics for industrial applications. He is an author of numerous papers and books aiming at practical applications in industrial environments. He was a Reviewer and a Committee Member with DAAD and in different international journals, he is actively participating in new developments in science and engineering. He is a member of SIAM and AMS, USA. He was the recipient of different awards from the Hans Voith Foundation, an award of The Baden- Wuerttemberg Employers Association of the Metal Industry e. V., Germany, an award of the University Association Ulm, a Scholar Grant by Wolfram Research, USA, and the Merckle Research Award, Germany.

M.Sc. Eng. Hussein Fawzy. Eng. Hussein held positions at different academic and industrial institutions during his career. In the academic institutions, he developed multiple sensor systems to support the educational activities. For the industry, Eng. Hussein held the position for Robotics Director at two different companies with different nationalities (USA and Egypt). In these positions, he led teams to develop multiple commercial robotics and automation systems. Graduating from the German University in Cairo from the department of Mechatronics, Eng. Hussein was appointed to the Industrial Activities Chair of the IEEE Egypt Section – RAS Chapter (recipient of the 2015 IEEE RAS Chapter of the Year Award and 2012 Chapter of the Year Award in IEEE Region 8). He was also a member of the judging committee for EED 2017 organized by IEEE Egypt Section, IEEE Young Professionals (YP) Egypt and the Egyptian Engineering Day (EED) organizing committee. Eng. Hussein received his M.Sc. degree from the German University in Cairo in 2014. His work focused on the frequency and vibrations analysis of different industrial systems for early diagnosis and prediction of failures. He is currently working on his Ph.D. at the German University in Cairo in the field of robotics.