POLITECNICO DI TORINO
Repository ISTITUZIONALE

A 3D Path Planning Algorithm based on PSO for Autonomous UAVs Navigation: 9th International Conference, BIOMA 2020, Brussels, Belgium, November 19–20, 2020,

Original
A 3D Path Planning Algorithm based on PSO for Autonomous UAVs Navigation: 9th International Conference, BIOMA 2020, Brussels, Belgium, November 19–20, 2020, Proceedings / Mirshamsi, A., Godio, S., Nobakhti, A., Primatesta, S., Dovis, F., Guglieri, G. (THEORETICAL COMPUTER SCIENCE AND GENERAL ISSUES). - In: Bioinspired Optimization Methods and Their Applications / Mirshamsi A., Godio S., Nobakhti A., Primatesta S. , Dovis F, and Guglieri G.. - ELETTRONICO. - [s.l] : Springer, 2020. - ISBN 978-3-030-63709-5. [10.1007/978-3-030-63710-1]

Availability:
This version is available at: 11583/2850165 since: 2020-11-02T10:20:15Z

Publisher:
Springer

Published
DOI:10.1007/978-3-030-63710-1

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature’s AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/978-3-030-63710-1

(Article begins on next page)

18 November 2026

A 3D Path Planning Algorithm based on PSO for Autonomous UAVs Navigation

Alireza Mirshamsi[1[0000-0002-1520-1791]], Simone Godio[2[0000-0002-4474-8454]], Amin Nobakhti[3[0000-0002-8495-0809]], Stefano Primatesta[2[0000-0001-7903-5019]], Fabio Dovis[3[0000-0001-6078-9099]], and Giorgio Guglieri[2]

[1] Sharif University of Technology, Tehran, Iran
Department of Electrical Engineering
alireza.mirshamsi@ee.sharif.ir, nobakhti@sharif.ir
https://www.en.sharif.edu

[2] Politecnico di Torino, Turin, Italy
Department of Mechanical and Aerospace Engineering
simone.godio@polito.it, stefano.primatesta@polito.it,
giorgio.guglieri@polito.it
https://www.dimeas.polito.it

[3] Politecnico di Torino, Turin, Italy
Department of Electronics and Telecommunications
fabio.dovis@polito.it
https://www.det.polito.it

Abstract. In this paper, a new three-dimensional path planning approach with obstacle avoidance for UAVs is proposed. The aim is to provide a computationally-fast on-board sub-optimal solution for collision-free path planning in static environments. The optimal 3D path is an NP (non-deterministic polynomial-time) hard problem which may be solved numerically by global optimization algorithms such as the Particle Swarm Optimization (PSO). Application of PSO to the 3D path planning class of problems faces typical challenges such slow convergence rate. It is shown that the performance may be improved markedly by implementing a novel parallel approach and incorporation of new termination conditions. Moreover, the exploration and exploitation parameters are optimized to find a reasonably short, smooth, and safe path connecting the way-points. As an additional precaution to avoid collisions, obstacle dimensions are artificially slightly enlarged. To verify the robustness of the algorithm, several simulations are carried out by varying the number of obstacles, their volume and location in space. A certain number of simulations exploiting the random nature of PSO are performed to highlight the computational efficiency, and the robustness of this new approach.

Keywords: Particle Swarm Optimization (PSO) · 3D path planning algorithm · unmanned aerial vehicle (UAV) · Autonomous Navigation.

1 Introduction and related works

The problem of path planning in the presence of obstacles is one of the cornerstones of autonomous UAVs navigation. In all those critical scenarios where it is necessary to act quickly, such as earthquake-stricken areas, quarries, crevasses, or in others GPS denied/degraded environments, for UAVs the ability to process the path independently in a short time is critical. In [1], an interesting path planning solution is developed for urban environments based on MPC (Model Predictive Control) for a UAV rotary-wing fleet.

In 2007, a seminal contribution for real-time 2D path planning based on PSO in dynamic environments in the presence of circular obstacles was published [2]. PSO-based path planning algorithms were subsequently used in several studies such the one presented in [3] to solve path planning problems in complex 2D scenarios populated by a large number of articulated-shaped obstacles. In this work the PSO is used to optimize trajectories in terms of smoothing and path length. In [4] a path planning in 2D environments in limited survival time without obstacles is presented which aims to reduce the computational time associated with PSO. In subsequent years, further PSO-based approaches were developed, for 2D path planning with static or dynamic obstacles, in [2, 5–9] PSO based path planning for multi-robot applications is considered by [5] in which both collision-avoidance with obstacles, and also with trajectories of other units is considered simultaneously.

One of the early studies on 3D path planning is presented in [6]. This work is built on an analogy between trajectories and fluid lines around a body. Subsequently, PSO-based algorithms are developed in [7] and [8] in order to improve upon the computational time and trajectory optimization point of view. Despite this, the 3^{rd} dimension significantly increases the complexity of the algorithm and, for this reason the computational times are in the order of minutes or hours. As a consequence, only simplified environments are still considered.

Since algorithms of this nature (evolutionary algorithms) are useful to find the global minimum for a problem, the PSO is therefore effective to the search for trajectories of minimum length; and for this reason that it is adopted in the following discussion.

In this paper we propose an innovative approach to to find a global sub-optimal solution of a 3D path planning problem with a significant reduction in computational time, even in the presence of several obstacles. This, enables full autonomous UAVs navigation in several previously unattainable possibilities. Simulation results demonstrate that the proposed strategy is able to compute a sub-optimal solution with a computational time lower than 1 second. The main feature of the proposed algorithm in parallel implementation of the 3D path planning problem.

The paper is organized as follow. In Section 2, the classic particle swarm optimization is presented with the novel modifications developed in this work. The definition of the objective function for the path planning problem and the parameters tuning are also presented in this section. In Section 3, results are pre-

sented for 4 different environments of increasing complexity. Several simulations are considered by varying the starting point and the target.

2 Problem formulation

In this paper, the proposed PSO-based algorithm solves a path planning problem searching for the shortest path connecting a starting point to a target or sequence of targets. It is assumed that the precise offline map of the environment is available to the drone. Moreover, the drone is required to reach complete stop (zero-velocity) above each target, if there are more than one target to reach.

In this section, the standard PSO algorithm and the improvements proposed to have fast and reliable results for the problem of 3D path planning are presented. After the introduction of the objective function, the value of different parameters of the heuristic approach is presented. For each path, i.e. target i to target $i + 1$ we compute N_t points and a smooth and feasible flight path will be determined for the drone by considering spline interpolation between $N_{V\ ar}$ auxiliary points.

It is assumed that path planning is performed in a bounded space: the minimum and maximum of the positions in each direction in 3D space are determined based on the environment where the UAV flies. These are defined as Min_p and Max_p, where p could be x, y, or z. For stability of the algorithm, a boundary for the velocity of particles is needed: V_{min_p} and V_{max_p} are defined in equations 1.

$$\begin{aligned} V_{max_p} &= a(Max_p - Min_p) \\ V_{min_p} &= -V_{max_p}, \end{aligned} \tag{1}$$

where a is a tuning parameter.

2.1 Particle Swarm Optimization

The origin of this algorithm takes its cue from the study of the social behavior of a bird flock or a fish school by Berhart and Kennedy [9, 10] . It solves optimization problems by introducing a population of candidate solutions, called particles, and iteratively trying to improve each of them in relation to an objective function. In this study, this function is represented by a combination of smoothness, shortness, and safety of the proposed flight path. For the i^{th} particle of the swarm, the position and the velocity vector in the current and in the following time instant are defined as:

$$\begin{aligned} V_i^{k+1} &= wV_k^{\ k} + r_1 a\,(\overrightarrow{Pb}i^k - x_i^k) + r_2 c\,(\overrightarrow{Gb}i^k - x_i^k) \\ X_i^{k+1} &= X_i^{\ k} + V_i^{k+1} \end{aligned} \tag{2}$$

where the particle velocity is defined by the sum of the inertial contribution, the cognitive contribution and the global one with their respective speed values:

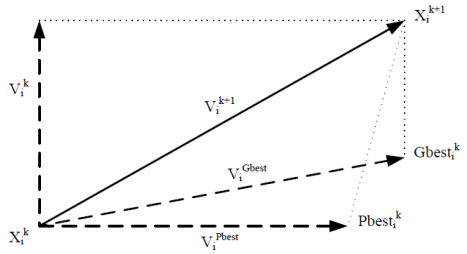


Fig. 1: PSO search mechanism in multidimensional search space, [11]

V_i^k, $\overrightarrow{Pbi^k}$ and $\overrightarrow{Gbi^k}$. While, for the calculation of the position of the particle at instant $k + 1$ the pose at instant k and the velocity at instant $k + 1$ are added. What is more, ω denotes the inertia weight, c_1 and c_2 are the personal and global learning constants respectively, r_1 and r_2 random values in [0,1]. Fig. 1 shows a schematic of equation 2.

2.2 Improvement with respect to standard PSO algorithm

Due to the slow convergence of the standard PSO algorithm for intensive problems such as 3D path planning, it is essential to tune and change the parameters of the standard algorithm to obtain satisfactory results. In this paper, minor changes in the standard algorithm itself are performed, and the parameters of PSO for the 3D path planning problem in an obstructed environment are appropriately tuned. Algorithm 1 shows the pseudocode of our PSO algorithm. Main features of our proposed algorithm versus the standard PSO algorithm is as follows:

1. Implementing parallel computing to find a sub-optimal path between different targets. In the standard PSO the best path between multiple targets is found by increasing the number of variables. However, this leads to costly computations and poor results. By proposing and successfully implementing a parallel form of path-planning we show that an efficient and accurate path can be found by multiple instances of parallel PSO with low number of variables and a low computational cost.
2. Parallel computing for each direction, i.e. x, y and, z, which leads to fast convergence in each direction with a low computational cost.
3. Control of the velocity of particles to remain within the permitted range. When these ranges aren't respected, to obtain reliable results, it is wise not only to saturate the magnitude of velocities, but also using velocity mirroring, which guarantee the particles to stay in the right path and to reach positions with lower cost in less time.
4. Considering 3 distinct stop conditions including, maximum number of iterations, obtaining reduced cost less than γ % in N_γ consecutive iterations, and, finding a path which its length is equal to K_L times minimum path

length possible, where minimum path length possible is direct line between the starting point to destination point. Note that, γ, N_γ, and, K_L are tunable parameters, which will define in tuning section.

Algorithm 1: Proposed PSO algorithm for the problem of 3D path planning

```
%% initialization
Generate particle individuals with these structures; position, velocity, cost,
  bestPosition and bestCost;
Set positions of the particles randomly and velocities equal to zero and
  bestPosition equal to position;
Set costs of the particles by evaluating positions based on cost function and
  bestCost equal to cost;
Find global best position between these particles;
%% Main loop
Set IT = 0;
Set all(ActiveFlag) = true;
while any(ActiveFlag) is true do
    IT = IT + 1;
    for i = 1 : numel(targets) do
        if ActiveFlag(i) is true then
            for j = 1 : numel(particles) do
                for p = [x,y,z] do
                    Update velocities based on equation 2, and apply velocity
                      mirroring if velocity is out of range.
                    Update positions based on equation 2, and check they be in
                      the valid intervals.
                    Evaluate the costs of each position.
                    Update local best of each particle (bestPosition, bestCost)
                      and global best position and  cost.
                end
            end
            if any stop conditions has been satisfied then
                Set ActiveFlag(i) = false;
            end
        end
    end
end
```

2.3 Objective function

The objective function consists of two parts, one for path length and one for obstacle avoidance. Equation 3 illustrates the objective function to be minimized in our problem.

$$Cost = R + \beta V, \tag{3}$$

where R is the total length of the path, V indicates a violation of path defined in equation 4, more than 0 when the path crosses an obstacle and, β is the coefficient of the penalty part.

$$V = \sum_{i=1}^{N_o} \sum_{p=x,y,z} \max\left(\frac{\sum_{j=1}^{N_t} (R_{p_i} - |p(t_j) - O_{p_i}|)}{N_t}, 0\right), \tag{4}$$

where N_o is the number of obstacles, R_{p_i}, and O_{p_i} are respectively dimension and center of i^{th} obstacle corresponding to each direction, $x(t_j)$, $y(t_j)$, and $z(t_j)$ are the coordinates of the path at time t_j in each direction, and, N_t is the resolution of path over time. Note that, R_{p_i} is greater than actual dimension of obstacle corresponding to each direction, and $R_{p_i} = r_{p_i} + R_{Cons}$, where r_{p_i} ($p = x, y, z$) are actual dimension of obstacle and R_{Cons} is a conservative margin which is related to dimensions of the drone itself.

2.4 Parameter setting

To reach accurate and fast results, $a = 0.1$, $\beta = 200$, $\gamma = 1\%$, $N_Y = 5$, $N_{V\ ar} = 3$, $N_t = 100$, and, $K_L = 1.08$ are found by trail and error. For a small quad-copter, we consider $R_{Cons} = 0.4\ m$. The number of particles for this problem is 150 with maximum iterations of 50, i.e. the final solution should be reachable in less than or equal to 50 iterations.

For best performance, it is important to tune exploration and exploitation parameters, i.e. c_1 and c_2, correctly. Therefore, by simulating different conditions, we use the constriction coefficient introduced by Kennedy [12] based on equation 5 to tune c_1 and c_2.

$$\begin{aligned} &\varphi_1, \varphi_2 > 0, \varphi = \varphi_1 + \varphi_2 > 4 \\ &x = \frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \\ &c_1 = x\varphi_1 \\ &c_2 = x\varphi_2 \end{aligned} \tag{5}$$

where the optimal solution is $\varphi_1 = \varphi_2 = 2.05$, so $x = 0.7298$ and $c_1 = c_2 = 1.4962$. Therefore, the exploration and exploitation coefficients have a balance and lead to fast and robust results for the problem of 3D path planning. Moreover, we consider inertia wight as $w = w_{damp}x$, where w_{damp} is variable by iterations and as $w_{damp} = 0.99^{it}$. This approach has a significant role in reducing the computational time for our problem.

For what concerns the reliability of the algorithm, the following stopping criteria are fixed:

- Cost below γ% in N_γ consecutive iterations, where γ and N_γ are 1 and 5 respectively. This condition means that a feasible solution is obtained as there is not a strong reduction in the cost during the last consecutive iterations.
- A path length equal to K_L (1.08) times the minimum path length possible is obtained. Where minimum path length possible is represented by the direct line between the starting point to destination point.
- Maximum number of PSO iterations always below 100 (to limit the computational time).

3 Results and Considerations

In order to evaluate the efficiency of the algorithm in terms of computational time and path length, 4 different environments are built with increasing complexity. In all these 4 maps a fixed rectangular base parallelepiped control volume (CV) containing several obstacles is considered (length = 25 m, width = 11 m, height = 5 m), which represents also the limit within the path can be elaborated by the algorithm. Fig. 2 shows the testing environment with detailed the percentage of obstacles defined as V_{Obst}/V_{CV} %, with V_{Obst} the volume occupied by obstacles.

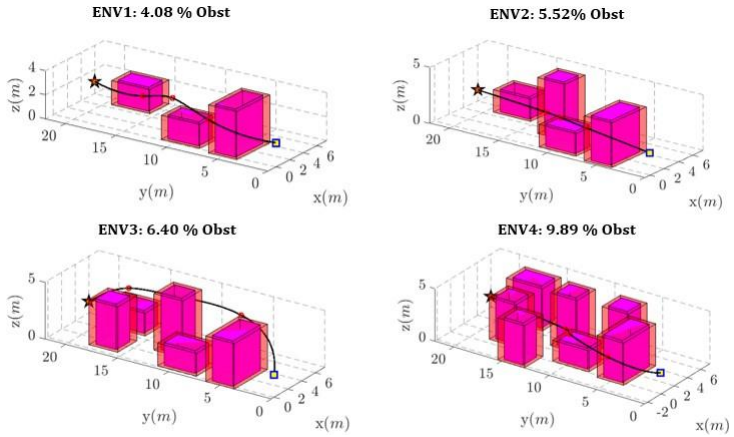


Fig. 2: MATLAB® simulated environments with their respective percentage of obstacles

3.1 Simulation in different environments

Due to the random nature of the PSO algorithm various runs with the same starting point ($[0;0;2]m$) and destination point ($[6;22;1.0]m$) are performed for each environment to test the robustness of the algorithm and its path length and computational time results in terms of variance and average value. In this

case, 50 runs for each environment are performed. Simulations are performed in MATLAB ® (R2020a) in a PC of Windows 10 OS, Intel(R) Core(TM) i7-7700 CPU with 2.80GHz and 16GB RAM.

In Fig. 3, the results for the first environment are shown; the path length has a moderate oscillation between the maximum and minimum value of approximately $1.3m$. The computational time, except for the initial outlier due to the environment setting, is quite stable and oscillate around 0.30 s.

In Fig. 4, results for the second environment are shown. The path length average value is varies negligibly. Instead, its variance starts growing and the max gap between the maximum and minimum value of approximately increase up to $2.3m$. While, the computational time trend and the average is not varied considerably.

Fig. 5 shows results for the 3rd environment where the percentage of obstacle increase up to 6.40 %. In the path length an important outlier manifests; while, the rest of the simulation results are stable and similar to the Fig. 4 results. In this case, the increased % of obstacles involves a rise in the computational time average, which increases to 0.45 s. Slight growth in the computational time variance can be noted too.

In the last environment a high degree of environment complexity is applied. Seven obstacles and 9.89% of obstruction make up the environment, as shown in Fig. 2. Fig. 6 reports an increase in the variance of path length and computational time. Also, an increase in the computational time average is registered, which settles around the still limited value of ~ 0.85 s.

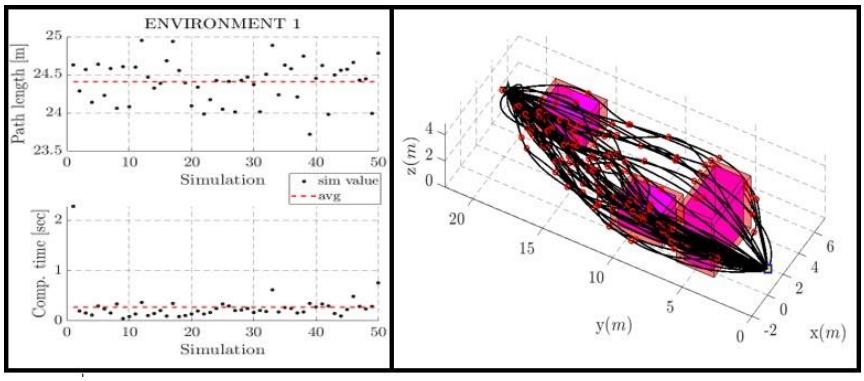


Fig. 3: Path length and computational time results for environment 1

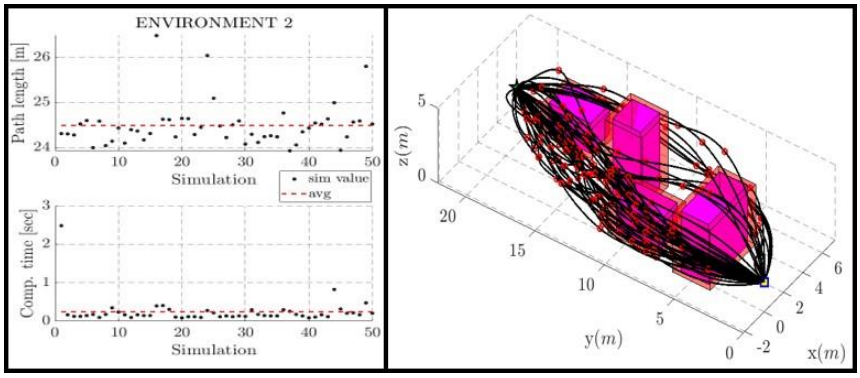


Fig. 4: Path length and computational time results for environment 2

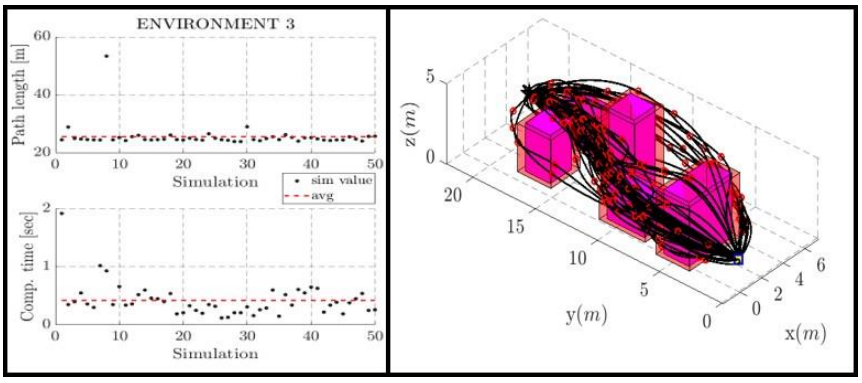


Fig. 5: Path length and computational time results for environment 3

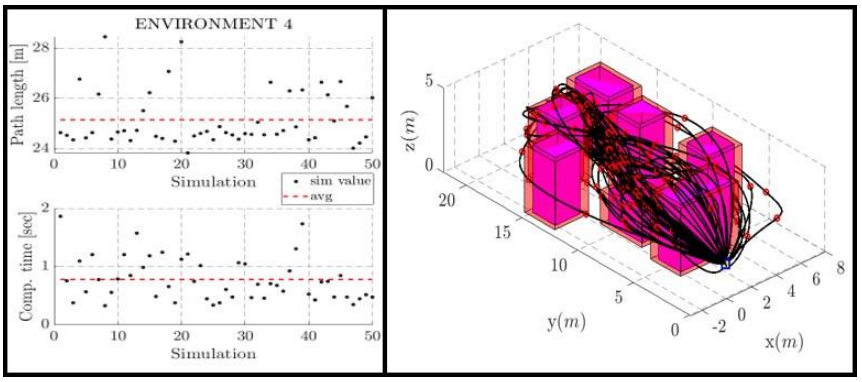


Fig. 6: Path length and computational time results for environment 4

To find the std of different parameters of the algorithm, 200 simulations for each environment were performed. The results are shown in Fig. 7. Simulations are performed in MATLAB ® (R2020a) in a PC of Windows 10 OS, Intel(R) Core(TM) i5-2400 CPU with 3.10GHz and 16GB RAM.

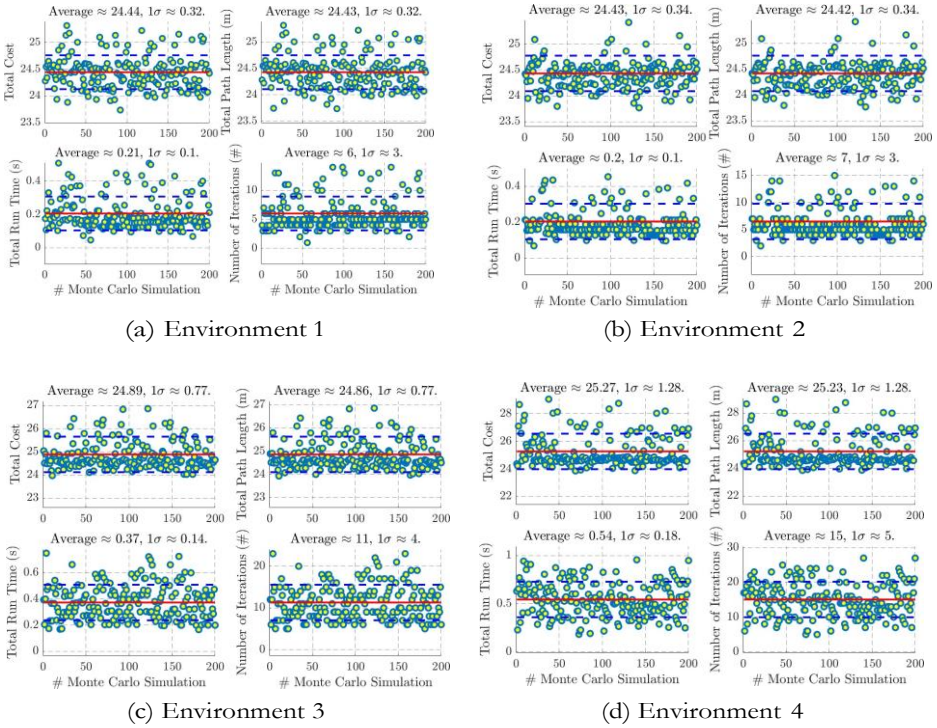


Fig. 7: Simulations results for the 4 different environments with 200 runs

In this case, the total cost and the number of iterations are plotted also. As previously shown, it is notable that a consequence of increasing complexity of the environment is to raise the standard deviation of each magnitude. Another consequence is the increase of the gap between the maximum path length and the minimum one; this problem can be overcome thanks to the considerably short computational time that allows elaborating different solutions every second and selection of the best. The discrepancy notable in the average computational time compared to previous results shown in Fig. 3, 4, 5, and 6, is mainly due to the use of different PCs in the two cases, as specified.

To resume, in the simplest environment there is a reduced standard deviation with the presence of a reduced number of outliers. As the complexity of the environment increases, the standard deviation rises and the presence of outliers consequently decreases.

In Fig. 8, the results are plotted in terms of average values of path length and computational time. As expected, the path length and the computational time mean values increase with the complexity of the environment simulated. But, the positive result is the reduced value of the derivative of both curves.

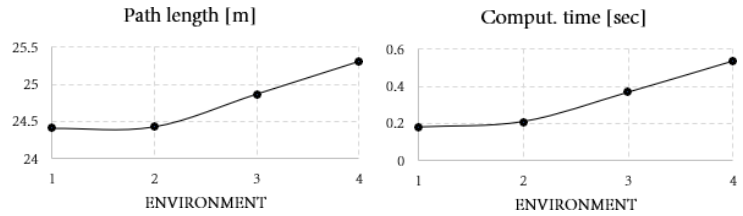
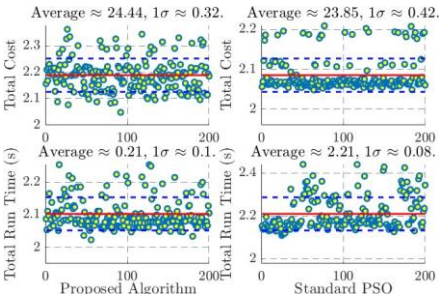
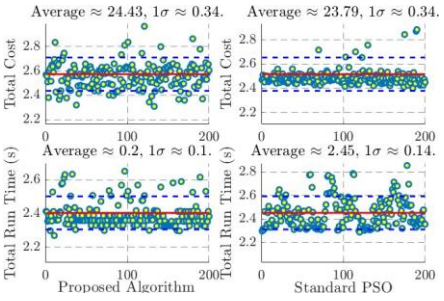


Fig. 8: Path length and computation time over more complex environment

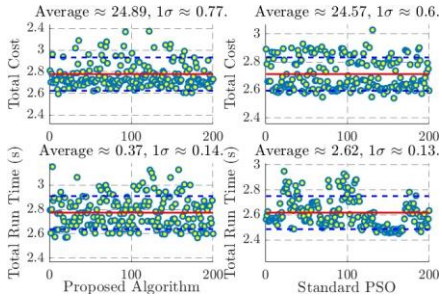
3.2 Comparison with standard PSO



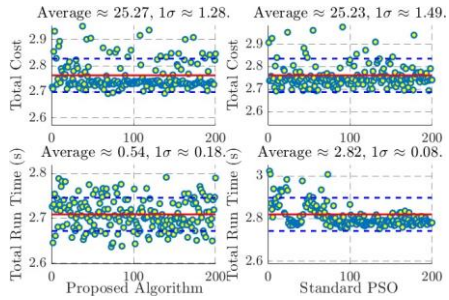
(a) Environment 1



(b) Environment 2



(c) Environment 3



(d) Environment 4

Fig. 9: Comparison results for the 4 different environments with 200 runs

To show the effectiveness of the proposed approach, a comparison for each environment with the standard PSO is performed. The differences between the proposed approach and standard PSO are 4 mentioned points in sec 2.2 and equation 5. In this comparisons, the standard PSO parameters are $c_1 = c_2 = 1.7$, and $w = 0.6$ based on [13]. Other common parameters are the same as proposed algorithm including the cost function in equation 3. Both algorithms are performed in MATLAB ® (R2020a) in a PC of Windows 10 OS, Intel(R) Core(TM) i5-2400 CPU with 3.10GHz and 16GB RAM.

As Fig. 9 shows the standard PSO has an lower total cost (less than 3%), but the total run time of the proposed algorithm is almost $6 \sim 10$ times faster.

3.3 Different starting and destination way-points

To further validate the algorithm other tests varying the starting and the destination way-points are performed. The aim is to elaborate shorter and more critical paths. In Fig. 10 the results obtained for the first environment are shown. It is notable that in all 50 runs of the three cases the algorithm converges in a sub-optimal collision-free path. The computational time is of the same magnitude as the results shown above and well below one second.

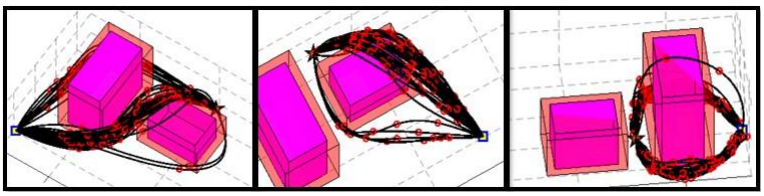


Fig. 10: Results for different starting and destination point in environment 1

The objective of this analysis, as anticipated, is to verify that the algorithm elaborated does not present divergences even when faced with trajectories in areas more populated by obstacles than those shown in Fig. 2. It is necessary to collect numerical data relative to the standard deviation of the obtained results because the stability reached by the algorithm from Fig. 10. In the latter is shown how the processed trajectories show a comparable length without anomalies or divergences of any type.

4 Conclusions and further developments

It is shown that even for the hardest scenario, the computational time always stays below one second with a stable sub-optimal path solution. This consistency in achieving stable fast sub-optimal path solution represents a marked improvement over previous algorithms such as [7] and [8]. In fact, as shown in Section

3, the computational time never exceeds 1 second, even in the most complex environment analyzed with a stable and reliable 3D path planning solution.

Moreover, since all the simulations run on MATLAB® for convenience; then the computational time can still be reduced by implementing the logic in an embedded platform with a lower level language code as C/C++ in on-board platforms. For further developments, other optimizations are in consideration to further reduce the calculation time and meet the real-time requirements without losing the quality of solution. Moreover, due to fast convergence of the proposed algorithm, it could be used for dynamic obstacles in real time. The final aim of this work is to provide the UAV with the ability of autonomous real-time path planning, for critical environments with a high percentage of obstacles also, which is one of those key aspects for autonomous flight in unknown GPS denied/degraded and critical environments in general.

References

[1] David Hyunchul Shim and Shankar Sastry. "A dynamic path generation method for a UAV swarm in the urban environment". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2008, p. 6836.

[2] Yanling Hao, Wei Zu, and Yuxin Zhao. "Real-time obstacle avoidance method based on polar coordination particle swarm optimization in dynamic environment". In: *2007 2nd IEEE conference on industrial electronics and applications*. IEEE. 2007, pp. 1612–1617.

[3] Ellips Masehian and Davoud Sedighizadeh. "A multi-objective PSO-based algorithm for robot path planning". In: *2010 IEEE International Conference on Industrial Technology*. IEEE. 2010, pp. 465–470.

[4] Harshal S Dewang, Prases K Mohanty, and Shubhasri Kundu. "A robust path planning for mobile robot using smart particle swarm optimization". In: *Procedia computer science* 133 (2018), pp. 290–297.

[5] Asma Ayari and Sadok Bouamama. "A new multiple robot path planning algorithm: dynamic distributed particle swarm optimization". In: *Robotics and biomimetics* 4.1 (2017), p. 8.

[6] Peng Yao, Honglun Wang, and Zikang Su. "UAV feasible path planning based on disturbed fluid and trajectory propagation". In: *Chinese Journal of Aeronautics* 28.4 (2015), pp. 1163–1177.

[7] Zhuang Shao et al. "Path planning for Multi-UAV formation rendezvous based on distributed cooperative particle swarm optimization". In: *Applied Sciences* 9.13 (2019), p. 2621.

[8] LIU Yang et al. "Collision free 4D path planning for multiple UAVs based on spatial refined voting mechanism and PSO approach". In: *Chinese Journal of Aeronautics* 32.6 (2019), pp. 1504–1519.

[9] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[10] Russell Eberhart and James Kennedy. “A new optimizer using particle swarm theory”. In: *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Ieee. 1995, pp. 39–43.

[11] Mahamad Nabab Alam. “Particle swarm optimization: Algorithm and its codes in matlab”. In: *ResearchGate* (2016), pp. 1–10.

[12] Maurice Clerc and James Kennedy. “The particle swarm-explosion, stability, and convergence in a multidimensional complex space”. In: *IEEE transactions on Evolutionary Computation* 6.1 (2002), pp. 58–73.

[13] Wei Zhang et al. “A simple way for parameter selection of standard particle swarm optimization”. In: *International Conference on Artificial Intelligence and Computational Intelligence*. Springer. 2011, pp. 436–443.