Contemporary Engineering Sciences, Vol. 10, 2017, no. 32, 1571 - 1579
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2017.711193

Improvements of a Three-Dimensional Trajectory Planning Algorithm for UAV Navigation Based on Potential Fields

Leonardo Ramírez Alberto

Universidad Pedagógica Nacional, Bogotá, Colombia

Carlos Eduardo Cabrera Ardila

Universidad Pedagógica Nacional, Bogotá, Colombia

Diego Mauricio Rivera Pinzón

Universidad Pedagógica Nacional, Bogotá, Colombia

Abstract

Path planning is a subject of wide interest in applications for unmanned aerial vehicles. A great variety of solutions have been generated to find paths free of collisions in three-dimensional scenarios, among them, the algorithm based on independent mobile points MBPF which uses the concept of artificial potential fields for the evasion of obstacles, is an algorithm that can be used in both 2D and 3D environments with static or dynamic obstacles and manages to avoid falling into local minimums. MBPF can find trajectories in a variety of complex 3D scenarios. However, it can be improved so that the trajectory reduces its distance, smooth the route and in some cases, converge. To improve the response of the algorithm, 3 improvements are proposed, one based on a modification to the original algorithm and another two based on a post processing of the result. the first improvement directly modifies the initial distribution of the starting points of the algorithm by densifying the points in the areas close to the obstacles, the two remaining improvements are made by post-processing the trajectory obtained for which Visibility Graphs are used and Active Contours. As a result, a considerable reduction

in the distance to be traveled is evidenced in the final trajectory and a greater smoothness is perceived in the curves.

Keywords: Path Planning, Potential Field, Active Contours, Visibility Graphs

1 Introduction

UAVs are unmanned or remotely controlled aerial vehicles, used for various applications in the industry [4]. According to [5] the UAV are classified into three different categories, fixed wing, flapping wing and rotary wing, within this last category are the quadrotors, which are suitable for the application of algorithms that generate trajectories, due to their characteristics of easy control, low energy consumption, high flight safety and high maneuverability. For a quadrotor to perform missions autonomously requires algorithms for the planning of safe trajectories, this is a problem that has received considerable attention in recent years, proof of this are the recent studies on trajectories for UAV as they are those of [2], [3], [7], [8] among many others, such algorithms are efficient if they can obtain a solution, if any, to complex and disordered environments [1].

The MBPF technique allows generating trajectories free of collisions, applying artificial potential fields (APF) a navigation space is configured with sigmoid functions associated with each obstacle. Subsequently the MBPF algorithm traces a series of points that constitute the trajectory, these points will be affected by the potential field of each obstacle, moving by effects of the gradient to a Safe Zone. The concept of Safe Zone is defined by a scalar value that determines the approximation of the final path with the obstacle depending on the APF [6]. The MBPF algorithm can find trajectories in complex 3D scenarios. however, it has some weaknesses in certain specific conditions, for example, in the case that the safe zone is far from the obstacles the trajectory obtained could be disordered and intricate, and in certain cases where the obstacles configure concave shapes the trajectory will pass through the obstacle having an imminent collision. To improve the response of the algorithm, 3 improvements are proposed, one based on a modification to the original algorithm and another two based on post-processing to the result. The three proposed modifications are presented below, and as results, two representative cases are postulated to show the improvements generated, and finally the conclusions are presented.

2 Methodology

Densified-MBPF

The MBPF technique distributes the initial points in a uniform way, in certain configurations of the environment this could be counterproductive, especially in scenarios with large variations in its potential field, this can generate problems because if there are not enough points located in the areas of greater variation of the field the algorithm could generate longer routes or increase the risk of collision with obstacles. To avoid the problem described an algorithm that densifies the distribution of the original points is proposed, the idea is that there are more points in the areas of greater variation of the potential field, that is, around the obstacles.

To implement Densified MBPF, initially each point of the initial configuration is evaluated, if a point is in a non-collision zone (safe zone), the next point is evaluated, but if a point is in a collision zone, they begin to intensify the points in said area using equation 1.

$$(X, Y, Z) = (X_0, Y_{0,} Z_0) + \lambda * (V_x, V_y, V_z) \quad (1)$$

Where (X, Y, Z) are the coordinates of $point_{n+1}$, $(X_0, Y_{0,} Z_0)$ are the coordinates of $point_n$, (V_x, V_y, V_z) is the vector direction between the $point_n$ and $point_{n+1}$, and λ is the sensitivity coefficient that allows to determine the amount of point intensification in the collision areas. The algorithm for the implementation is presented in Pseudocode 1. Figure 1 shows the densification result and the trajectory generated with that initial configuration.

Pseudocode 1. Densified MBPF

```
Requires: Trajectory points P of MBPF
Requires: SafeZoneValue of trajectory T of MBPF
Var float: λ, m, sensibility, VectorDirection
Var out: NewXYZ
sensibility
m = 1
repeat
   VectorDirection ← [Point(m+l) − Point(m)]
   λ= 0
   while λ < 1 do
   (Xn,Yn,Zm) =Point(m) + λ* VectorDirection
   if PotencialField(Xn,Yn,Zn) > SafeZoneValue then
      NewXYZ={Xn,Yn,Zn)
   End_if
   η = η + sensibility
   End_while
End
```

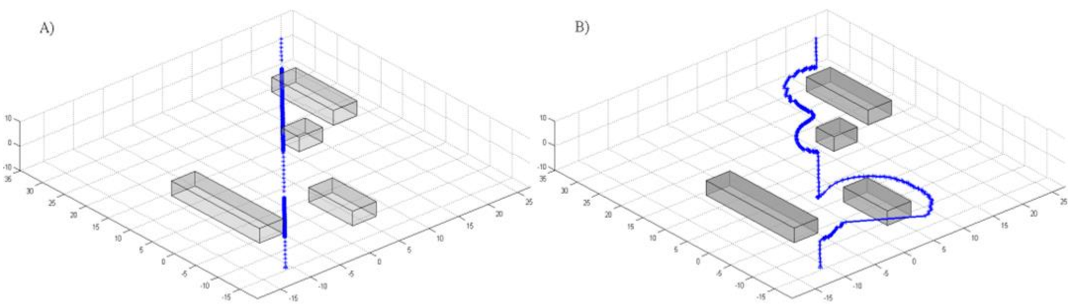

Figure 1: A) densification of points in the collision areas, B) Path generated by the MBPF

Visibility Maps MBPF

For the implementation of this improvement, the concept of visibility graphs is used as a principle, which consists in making a trajectory using the vertices of obstacles as points of the trajectory in a Euclidean space, in this way it is possible to avoid obstacles [3]. The Visibility Maps Algorithm consists of evaluating two distant nodes of the final path generated by the MBPF, measuring whether there is a risk of collision between the two nodes. If there is a collision risk, it will be evaluated with a node of less distance, and if there is no collision risk, these two points will be joined as part of the trajectory. With this strategy the distance of the trajectory can be reduced considerably. To determine if there is a collision between the two distant nodes, the parametric equations of the line are used according to equation 2.

$$\left(X_f, Y_f, Z_f\right) = \left(X_i, Y_{i,} Z_i\right) + \eta * \left(V_x, V_y, V_z\right) \tag{2}$$

Where$\left(X_f, Y_f, Z_f\right)$ are the coordinates of the furthest point, (X_i, Y_i, Z_i) are the coordinates of the current point, $\left(V_x, V_y, V_z\right)$ is the direction between the furthest point and the current point, and η is the coefficient that allows evaluating several points that are contained in the line current-point and far-point. The procedure is presented in pseudocode 2. An example of the result obtained with the Visibility Maps technique can be seen in figure 2.

Pseudocode 2. VisibilityMaps-MBPF

Require: Vectors x, y, z of trajectory **T of MBPF**
Require: SafeZoneValue of trajectory **T of MBPF**
Var: η, VectorDirection, n, m, Xn, Yn, Zn, *euclideanDistance*
Var out: Newx,Newy,Newz

```
n=length(x)
m = 1
repeat
   VectorDireccion ← [x(n), y(n), z(n)] - [x(m), y(m), z (m)]
   η = 0
   While PotentialField(Xn,Yn,Zn) < SafeZoneValue  && η < 1 do
        Xn = x(m)+ η* VectorDirection
        Yn = y(m)+ η * VectorDirection
        Zn = z(m) + η * VectorDirection
        η = η + euclideanDistance
  End_while
  if n ≥ 1 then
       Newx = Xn
       Newy = Yn
       Newz = Zn
  else
       η = η − 1
  End_if
End
```

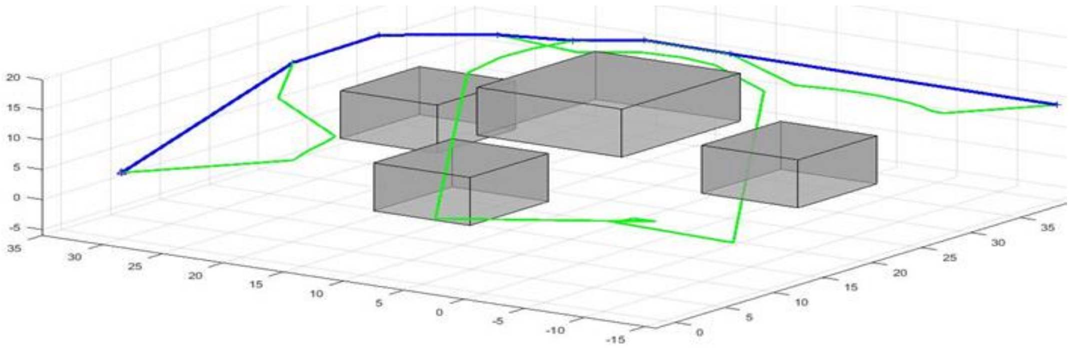

Figure 2: Comparison of the trajectories MBPF (green) vs visibility maps – MBPF (blue)

Snake MBPF

The Active Contours are used as a post-processing technique to improve the performance of the MBPF algorithm. To adapt the algorithm of active contours in this case only the internal energy of the Snake is required, ignoring the external energy, this is because the operation of the MBPF already considers the gradient of the obstacles. For the use of the technique, equation 3 is used.

$$E_{snakeMBPF}v(s) = \sum_{i=1}^{n-1} \alpha\left|\left|v_{i+1} - v_i\right|\right|^2 + \beta\left|\left|v_{i+1} - 2v_i + v_{i-1}\right|\right|^2 \quad (3)$$

Where n is the number of points in the MBPF path, $i = \{1,2,3,\ldots,n-1\}$ because the starting point ($i = 0$) and the ending point ($i = n$) are fixed points. Now, for each point to be evaluated, a neighborhood of movement possibilities must be created, for the case of 2D it will be a 3x3 matrix and in 3D of 3x3x3, with the intention that it can be evaluated and find the position with less energy, considering the restriction of the Safe Zone given by MBPF. The algorithm for the implementation is presented in the pseudocode 3. The result of the algorithm is evidenced in figure 3, which shows a considerable reduction of the distance and smoothing of the initial trajectory.

Pseudocode 3. Snake-MBPF

Require: Vectors *x*, *y*, *z* of trajectory **T** of **MBPF**
Require: Points n of trajectory **T** of **MBPF**
Require: SafeZoneValue of trajectory **T** of **MBPF**
Var: alpha, beta, proximity, variation, j, k, position, finalValue
Var: initialTrajectory

```
repeat
initialTrajectory ← x,y,z
 for k←2; n-1; k ← k+1
   Matrix
     for j ←1; n-1; j ← j+1
        internalEnergy (j) = position(j) of matrix
     End_for
```

```
      position ← j of less internalEnergy (j)
        while potentialField at Xi(position) Yi(position) Zi(position) > SafeZoneValue
          discard position
          Position ← j of next less internalEnergy (j)
        End_while
      x(k) ← Xi(position)
      y(k) ← Yi(position)
      z(k) ← Zi(position)
    End_for
      if initialTrajectory == x(k), y(k), z(k)
        Proximity = proximity - variation
      end_if
      if proximity < finalValue
        return
      End_if
```

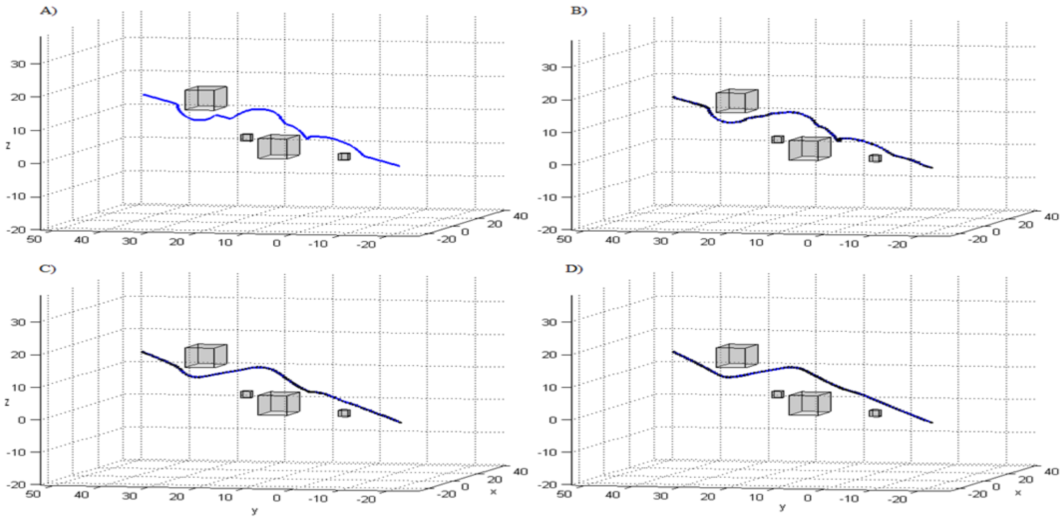

Figure 3: Sequence of the Snake-MBPF algorithm, A) the original MBPF path is shown, B) the Snake-MBPF in iteration 19, C) the Snake-MBPF in iteration 81, D) the MBPF-snake path completed in iteration 152

3 Results

For the evaluation of the algorithms two scenarios were developed which are representative in relation to the obtained improvements.

Case 1: For case 1 a selected scenario is presented so that a trajectory generated by the MBPF technique is observed, which converges, but it is likely to be improved. Case 1 is represented in figure 4, the original trajectory generated by MBPF is presented in green, the two post-processing Snake-MBPF (color cyan) and Visibility Maps-MBPF (blue color), by contrasting the results in the same axes are presented trajectories with shorter distance and more reliable for a UAV.

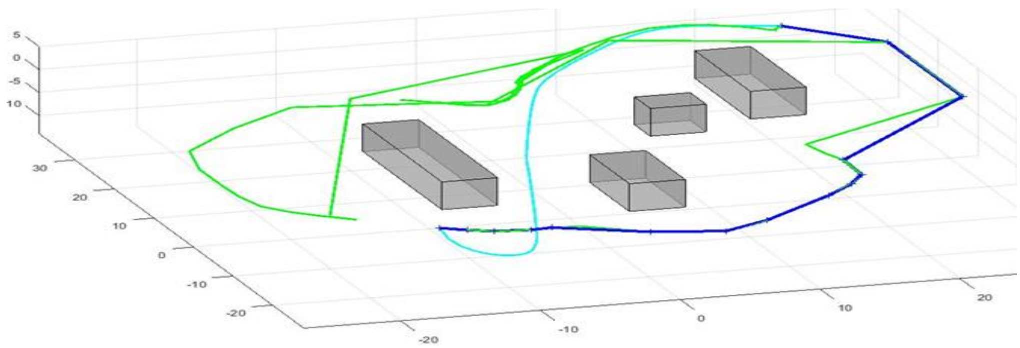

Figure 4: Comparison of techniques: MBPF (green), Snake-MBPF (cyan), visibility maps-MBPF (blue)

Case 2: For the second case, a scenario is presented with a trajectory generated by the MBPF technique that converges, but generates a collision with obstacles. Figure 5 illustrates the selected case, the original trajectory presented in green collides with the obstacle, in contrast, the Snake-MBPF techniques (cyan color) and Visibility Maps-MBPF (blue color), manage to converge and simultaneously reduce the distance for the navigation of a UAV.
Table 1 shows the comparison between the different strategies, Snake-MBPF and Visibility Maps-MBPF with respect to the original MBPF technique. The obvious improvement in all aspects is evident.

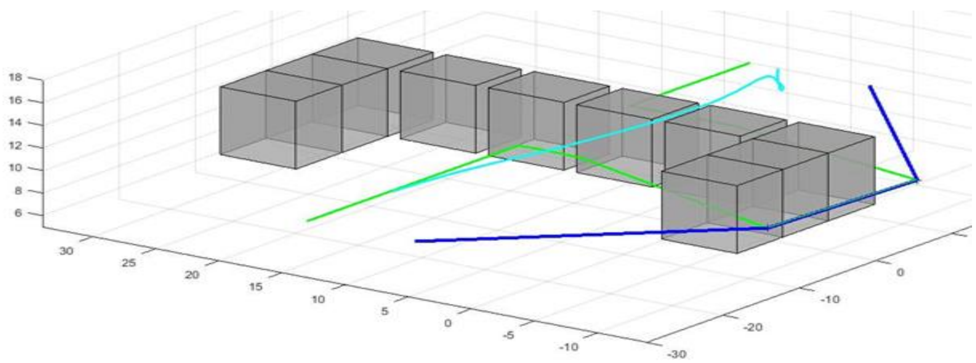

Figure 5: Comparison of techniques: MBPF (green), Snake-MBPF (cyan), visibility maps-MBPF (blue)

Table 1. Comparison of strategies

Criterion of evaluation	**Post-processing results**					
	Case 1			**Case 2**		
	MBPF	**Visibility Maps-MBPF**	**SnakeMBPF**	**MBPF**	**Visibility Maps-MBPF**	**SnakeMBPF**
Convergence	Yes	Yes	Yes	No	Yes	Yes
Distance	344 2193	95 2039	84 9004	137 5110	98 1735	89 6038
Percentage of distance reduction compared to MBPF	-	72.35%	75.34%	-	28.60%	34.83%
SafeZone value	0.0005	0.0005	0.0005	0.05	0.05	0.05
Number of points in the trajectory	100	100	100	255	255	255

Conclusions

The MBPF algorithm can find trajectories in complex 3D environments. however, it has some weaknesses in certain specific conditions, for example, for certain obstacle configurations the trajectory obtained can be disordered and intricate. To improve the results, 3 strategies were proposed, one based on a modification to the original algorithm and another two based on a post processing to the resulting trajectory.

The Densified-MBPF algorithm allows to generate more points on the trajectory in the collision zones, obtaining a trajectory with better resolution than the pure MBPF, achieving greater smoothness near the obstacles, better distribution of the points, intensifying the points only in the areas of collision. In some cases, the improvement is so important that it allows to achieve the convergence of the trajectory. The two post-processing to the resulting trajectories that are proposed are based on visibility maps and active contours. Favorable results are presented in terms of improving travel distances, and can be implemented in other similar trajectory planning techniques since they reduce the total trajectory, smooth the route, and reduce steep changes in direction.

References

[1] K. Baizid, R. Chellali, R. Luza, B. Vitezslav, F. Arrichiello, RRS: Rapidly-Exploring Random Snakes a New Method for Mobile Robot Path Planning, Chapter in *Intelligent Autonomous Systems 13*, Springer, Cham, 2016, 291-305. https://doi.org/10.1007/978-3-319-08338-4_22

[2] D. Kim, H. Kim and K. Huh, Local trajectory planning and control for autonomous vehicles using the adaptive potential field, *IEEE Conference on Control Technology and Applications*, (2017), 987-993.

https://doi.org/10.1109/ccta.2017.8062588

[3] P. Maini, P. Sujit, Path Planning for a UAV with Kinematic Constraints in the presence of Polygonal Obstacles, *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, (2016), 62-67. https://doi.org/10.1109/icuas.2016.7502625

[4] F. Nex, F. Remondino, UAV for 3d mapping applications: a review, *Appl Geomat.*, **6** (2014), no. 1, 1–15. https://doi.org/10.1007/s12518-013-0120-x

[5] S. Norouzi Ghazbi, Y. Aghli, M. Alimohammadi, A.A. Akbari, Quadrotors Unmanned Aerial Vehicles: A Review, *International Journal on Smart Sensing & Intelligent Systems*, (2016). 309-333.

[6] D.M. Rivera Pinzón, F.A. Prieto, R. Ramirez, Trajectory Planning for UAVs in 3D Environments Using a Moving Band in Potential Sigmoid Fields, *Brazilian Robotics Symposium and Latin American Robotics Symposium (SBR-LARS)*, (2012), 115-119. https://doi.org/10.1109/sbr-lars.2012.26

[7] D.M. Rivera Pinzón, F.A. Prieto-Ortiz, Planeación de trayectorias para cuadricópteros en ambientes dinámicos tridimensionales, *TecnoLógicas*, **18** (2015), 37-50. https://doi.org/10.22430/22565337.221

[8] Z. Zhang, J. Li, J. Wang, Sequential Convex Programming for Nonlinear Optimal Control Problem in UAV Path Planning, *American Control Conference*, (2017). https://doi.org/10.23919/acc.2017.7963240

Received: December 5, 2017; Published: December 20, 2017