Receding Horizon UAV Path Planning Via Gradient-Based Optimization of Ferguson Splines

Kyle Ingersoll, Patrick DeFranco, Bryce Ingersoll

April 10, 2015

***Abstract*—Path planning is an integral task of many unmanned air vehicle (UAV) applications. Minimizing travel time while avoiding obstacle collision is an important objective of almost any UAV flight path. In this paper, we seek to minimize the path length from an initial position to a final destination while avoiding collision with static, circular obstacles. We pose the path planning task as an optimization problem inside a receding horizon framework. After each planning step, the first segment of the path is then traversed and a new path is planned; this procedure continues until the final destination is achieved. The objective of the optimization is to minimize the distance to the final destination; the objective function also includes a term that promotes smooth, navigable trajectories. The path optimization is constrained by the obstacles and by a minimum and maximum step length. Path segments are modeled as Ferguson splines. The optimization of splines results in efficient, feasible paths even in complex obstacle fields. For particularly challenging scenarios, a multi-start approach is used to increase robustness.**

I. Introduction

Unmanned air vehicles (UAVs) have broad potential applications, including infrastructure monitoring, police surveillance, retail delivery, etc. Much of the current research surrounding UAVs focuses on achieving robust performance in a wide variety of environments. Examples of this type of research include GPS-denied navigation and sense-and-avoid, the latter being imperative for the safe integration of UAVs into the national airspace. In both of these specific areas, and in the field of robotics in general, path planning is an important task. Given a set of obstacles, path planning seeks to solve the problem of finding the best path to accomplish some objective, often getting from one point to another in the shortest time. Advanced path planning algorithms take into account factors other than just physical obstacles. These additional factors might include threat to the agent incurred over the path or the cumulative field of view of the agent as it traverses the path. In this paper, we consider the simpler path planning problem: finding the shortest path through an obstacle field.

A. Relevant literature

The UAV path planning problem has been approached in a myriad of ways. In [1], Beard and McLain present two widely referenced methods: Voronoi graphs and rapidly exploring random trees (RRT). The Voronoi graph method models each obstacle as a point and then partitions the area into a set of convex cells that each contain only one obstacle. The interior of each cell is closer to the obstacle contained in that cell than to any other obstacle. When applied to path planning, the edges of the cells can be traced to obtain a path through the obstacle field. Two immediately apparent shortcomings of this method are 1) its inability to model obstacles with nonzero area and 2) that it produces non-smooth paths. The first shortcoming can be overcome by modeling real obstacles with several point obstacles configured in the shape of the real obstacles. The second shortcoming can be compensated for by over-estimating the size of the obstacles; thus, when the UAV overshoots the desired path, there is some factor of safety that it will not collide with an obstacle. The second method, RRT, is an exploration algorithm that uniformly, but randomly, searches an area. When using RRTs for path planning, the algorithm checks for a feasible path from the branch ends to the final destination at the end of each iteration. When a feasible path is found, the algorithm then traverses the path and searches for possible links between non-consecutive path nodes, thus smoothing and shortening the path. The RRT method also produces non-smooth paths

that are inherently difficult for a UAV to follow precisely.

Other path planning approaches include probabilistic road maps (PRMs) [2]–[4], graph-based shortest path algorithms [5], and many other methods [6], [7]. Each planning scheme presented in the cited works have advantages and disadvantages in various situations. For a more thorough discussion, we refer the reader to the cited survey papers.

In this paper, we present a novel path planning algorithm that efficiently and robustly finds a feasible path through a complex obstacle field. Our approach integrates and improves on aspects of [8] and [9]–[11]. These papers frame the path planning problem in terms of optimization.

The authors of [8] use receding horizon control, in which the controller plans several short segments in successive steps instead of planning the complete path all at once. At each time step, the controller minimizes the time required to travel from the path segment's end point to the final destination, where the path segment is constrained by some planning horizon. The robot then precedes a given distance along that path and then repeats the search.

This receding horizon approach has the advantage of reducing a very complex problem into several smaller, more tractable problems. Additionally, since the controller periodically recalculates the path, it is more robust to errors, uncertainties, and unknowns in the system model or world map. It can also better handle obstacles that change in shape or position with time than a single-pass planner can.

A downside of receding horizon path planning is "entrapment," or its tendency to become stuck, particularly when facing obstacles with concave shapes [8]. As far as the planner can see, the best path is into the obstacle. After taking a step, it finds the concave interior of the obstacle and cannot proceed further.

Our work also builds on the work done for ground robots in [9]–[11], in which the authors optimize path segments described by Ferguson splines for ground robot navigation. We make a few key improvements in our work. First, the works described in the list papers optimize the trajectory from start to finish in a single pass. Our path planning problem involves a more complicated obstacle field that would be more difficult to plan in a single pass, so we implement a receding horizon approach. In addition, those papers use a penalty that is a function of the distance from the obstacles which consequently penalizes paths that may be a safe distance from obstacles. In our work, we implement the distance from obstacles as a constraint, thus allowing the path to go right to the border of the obstacles. This allows for shorter paths that are still feasible.

We choose to use a multi-start gradient-based method for our optimization rather than particle swarm optimization (PSO) or artificial bee colony (ABC) optimization. Gradient-based optimization quickly finds valids paths and we show the multi-start approach still avoids being trapped in local minima of the obstacle field.

B. Our Implementation

Our algorithm uses the receding horizon framework of [8] in combination with the Ferguson spline approach of [9]. We design our objective function to favor paths that are navigable by a typical fixed-wing UAV. Section II defines the problem statement and simulation environment. Section III describes our proposed method. Section IV presents several paths produced by our method. Finally, Section V presents concluding remarks and potential future research directions.

II. Problem Statement

We seek to find a feasible and short path through an arbitrary obstacle field. The beginning location of the path is at $(0, 0)$ and the final destination is at $(100, 100)$. In the space $(x, y) \in [5, 95]$, we place n_{obs} obstacles; this ensures that the beginning and ending locations lie in the feasible region. The obstacle locations are selected using Latin hypercube sampling (LHS). Two sets of $n_{\text{obs}}/2$ obstacles are overlayed with successive calls of LHS. The design of the obstacle field is meant to ensure the obstacles are relatively evenly distributed but to also allow for the challenges of overlapping obstacles and concavities in the obstacle field. This type of field is meant to be representative of an inner-city or forest environment. The obstacles are circular with radii uniformly ranging from $r_{\text{min}} = 3$ to $r_{\text{max}} = 7$. Non-circular obstacles can either be conservatively modeled with a single circular obstacle or more closely modeled with several smaller circular obstacles. A given randomly generated obstacle field

may contain concavities; a good path planner should be reasonably robust to these types of challenges.

We also seek to plan a path that could be followed by a typical fixed-wing UAV. In our UAV model, we assume the simple case of level flight with no wind. Thus, only turning dynamics are considered. We further ignore side slip and assume that all turns are coordinated turns. A coordinated turn is described by

$$\dot{\chi} = \frac{g}{V_g} \tan\phi \cos(\chi - \psi),$$

where χ is the aircraft course angle, ψ is the aircraft heading, ϕ is the roll angle, g is the gravitational constant, and V_g is the aircraft ground speed. The turn radius of the aircraft is described by

$$R = \frac{V_g \cos\gamma}{\dot{\chi}},$$

where γ is the flight path angle (vertical climb). In level flight $\gamma = 0$, and with no wind or side slip, $\chi = \psi$. Combining the equations and making the mentioned assumptions, the turn radius R is defined by

$$R = \frac{V_g^2}{g \tan\phi}. \tag{1}$$

III. Our Method

In this section, we describe the overall framework of our approach (Section III-A) and provide a detailed formulation of the optimization problem we seek to solve (Section III-C).

A. Overall Framework

Our path planner uses a receding horizon framework. We start by planning the first three path segments. The UAV traverses the first segment and then plans three more segments. This process is repeated until the end of the third segment is within some small gate distance of destination. When this occurs, the final three path segments are re-optimized to minimize their total length, and the path planning is complete.

Each path segment is modeled as a Ferguson spline. Ferguson splines are described by

$$X(t) = P_0 F_1(t) + P_1 F_2(t) + P_0' F_3(t) + P_1' F_4(t)$$

where the basis functions are given by

$$\begin{aligned} F_1(t) &= 2t^3 - 3t^2 + 1 \\ F_2(t) &= -2t^3 + 3t^2 \\ F_3(t) &= t^3 - 2t^2 + t \\ F_4(t) &= t^3 - t^2, \end{aligned}$$

where P_0 and P_1 are the beginning and end points of the spline and P_0' and P_1' are the derivatives of the spline at these points. In our application, $t \in [0, 1]$.

As defined in [12], Ferguson splines can have one of any three boundary conditions. In this work, we choose the constraint of having defined derivatives at the boundaries. Having the derivatives of the end points be explicit spline parameters is a major advantage of using Ferguson splines as it allows the whole path to be smooth path by implicitly requiring derivatives to match at the spline joints.

B. Vehicle Dynamics

A disadvantage of Ferguson splines is that it is difficult to analytically find the maximum curvature along a given spline, which means it is difficult to understand feasibility of the path for a given set of vehicle dynamics. We experimented with cubic and quadratic Bezier curves to model the path segments and considered using other approaches, such as fixed radius circles or many discrete line segments. We decided that it was more valuable to continue using Ferguson splines instead of other path definitions; although those methods could have made defining a minimum turn radius more straightforward, they would have been more difficult to control or would have increased the complexity of the optimization problem.

Due to the nature of our objective function (see Section III-C2), paths generated will generally wrap tightly around an obstacle, move in a straight line between obstacles, or turn slightly in the direction of a better path that becomes apparent as the planning horizon progresses. With careful definition of the obstacles to meet a minimum radius and safety margin, the minimum turn radius constraint can still be met with reasonable assurance. For example, in our tests we have a minimum obstacle radius of 3 meters. Using Equation (1), we can calculate that with a roll angle of 60 degrees, the maximum airspeed in the tightest turn would be just over 7 meters per second, which is feasible for many small, foam-type UAVs. Such calculations could be run for

a given obstacle field to set feasible safety margins for obstacle size and to determine what type of vehicle can fly through a given field.

In considering vehicle navigation, we ignore the second order dynamics of the aircraft. Constraints on higher order dynamics would ensure that the change in turn radius does not exceed the roll capabilities of the aircraft. If the turns are smooth and the obstacles reasonably spaced, it is safe to assume that this constraint could be ignored, as was done in [13].

The dynamic constraints focus on fixed-wing UAVs. When using helicopter or multi-rotor UAVs, the turn constraints no longer apply. Like many ground robots, rotorcraft are able to move at arbitrarily low forward speeds and even rotate in place, thus allowing them to make a turn of any curvature.

C. Formulation of the Optimization Problem

1) Design Variables: Our optimization problem has 12 design variables: (P_{1_x}, P_{1_y}) and (P'_{1_x}, P'_{1_y}), the location and derivative of the first segment's end point, which is also the second segment's beginning point; (P_{2_x}, P_{2_y}) and (P'_{2_x}, P'_{2_y}), the location and derivative of the second segment's end point and the third segment's end beginning point; and (P_{3_x}, P_{3_y}) and (P'_{3_x}, P'_{3_y}), the location and derivative of the third segment's end point. The location and derivative of the first segment's beginning point is implicitly constrained to be the current location and derivative.

2) Objective Function: At each planning step, we minimize the objective function

$$f(x) = d\left(1 + \alpha \frac{\ell}{\ell_{\text{min}}}\right)$$

where

$$d = \sqrt{(x_{\text{final}} - P_{3_x})^2 + (y_{\text{final}} - P_{3_y})^2},$$

the distance between the end point of the third spline and the final destination. This objective function rewards 1) minimizing the distance to the final destination and 2) taking a direct path. The importance of these tasks is balanced by the scalar α. In our simulations, we set $\alpha = 0.2$.

The curvature reduction term

$$\alpha \frac{\ell}{\ell_{\text{min}}} \tag{2}$$

was used in [9]–[11]. This metric penalizes long paths that traverse only a short distance—that is, paths with high curvature. By tuning α, we can reach a good balance between straight paths and optimal ending points. A similar effect could be achieved by performing two optimization problems: first minimizing the distance to the final destination and then minimizing the path length given a fixed (P_{3_x}, P_{3_y}) computed with the first optimization problem. We believe that the single objective function is a more elegant approach. We note that with the objective function, there is no incentive to find a minimum curvature path when the third spline arrives at the final destination because $d = 0$ forces $f = 0$, regardless of curvature. In this case, we use two optimization steps to straighten the path.

3) Constraints: We impose constraints associated with the length of the path segments and with the obstacles. Each path segment must be longer than a minimum step size $P_{\ell_{\text{min}}}$ and shorter than a maximum step size $P_{\ell_{\text{max}}}$, set to 3 and 15, respectively, in our simulations. Enforcing a minimum step size helps the path escape from concavities in the obstacle field. Limiting the maximum step size enables faster solutions by reducing the design space. The length of a parametric curve is given by

$$\ell = \int_{\alpha}^{\beta} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}\, dt. \tag{3}$$

Unfortunately, when Equation (3) is applied to the Ferguson spline, the closed-form solution to the integral becomes difficult to compute analytically. Consequently, we estimate the length of each path segment by approximating the spline as a sum of small, straight elements,

$$\ell_{\text{est}} = \sum_{i=1}^{n} \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

where n equals 50. The step size constraints are then given as

$$P_{\ell_{\text{max}}} \geq \ell_{\text{est}}$$
$$P_{\ell_{\text{min}}} \leq \ell_{\text{est}}.$$

To ensure the obstacle constraints, we need to find the minimum distance between all obstacles and a path segment. Once again, the closed-form solution to this problem is difficult to compute analytically. Consequently, we sample each segment at n_s uniform intervals and require that each sampled point

along the spline meet the constraint requirement. In our simulations, $n_s = 30$, i.e. each segment is sampled at $t = 0.0345, 0.0690, \ldots, 0.9655, 1$. These constraints then take the form

$$r_i \leq \sqrt{(X_x(t) - x_{\text{obs}_i})^2 + (X_y(t) - y_{\text{obs}_i})^2}$$

where r_i and $(x_{\text{obs}_i}, y_{\text{obs}_i})$ are the radius and position of the i^{th} obstacle, respectively. We note that $t = 0$ must already be a feasible point. At each planning step, the optimizer only considers obstacles that might pose an active constraint in the current look-ahead window, i.e. obstacles that meet the requirement

$$\sqrt{(X_x(0) - x_{\text{obs}_i})^2 + (X_y(0) - y_{\text{obs}_i})^2} \geq 3P_{\ell_{\max}} + r_i.$$

Considering only the nearby obstacles improves the efficiency of the method.

4) Final Optimization Problem: As mentioned in Section III-C2, once the final destination has been achieved, we re-optimize the final three path segments. In this problem, we seek to minimize ℓ_{est}. We retain the constraints described in Section III-C3 except we consider all obstacles in the field rather than just those in the current look-ahead window to reduce code complexity.

5) Gradients: Gradients of both the objective function and constraint functions are computed using the complex step approximation [14] given by

$$\frac{\partial f}{\partial x_i} = \frac{\text{Im} f(x_i + j)}{h}$$

where $h = 10^{-30}$, and the complex variable j is only added to the i^{th} element of x. Though a more efficient gradient calculation method could be used to improve performance, the complex step method gives accurate gradients and was simple to implement. Exact gradients supplied by the complex step method result in quick and stable convergence.

6) Initial Guesses and Multi-Start: At the beginning of each planning step, the locations and derivatives of P_1, P_2, and P_3 are set to that of P_2, P_3, and P_3 from the previous planning step. Although this means that P_3 starts in an unfeasible region, it allows that point to freely expand into space guaranteed to be devoid of obstacles. It also generally improves time to convergence because the algorithm does not have to "redo the work" it performed in the previous planning step. Beginning with a random, feasible guess has been found to make the single-start approach more robust. However, it invariably increases time to convergence and designs longer paths.

When using a multi-start approach, the first guess is that described in the preceding paragraph. We then generate a random initial guess within the minimum and maximum step size radii. We then check that the random initial guess meets the step size and obstacle constraints; if the constraints are satisfied, the optimization problem is solved using that initial guess. If the constraints are violated, we continue to randomly generate initial guesses until a feasible guess is found or we reach the maximum number of iterations, usually set to 10. If the optimization routine converges to a solution, satisfies all constraints, and results in a lower objective function value than the previous minimum, that solution is retained. We perform the multi-start approach described above 10 times. Experimental results shows that multi-start adds considerable robustness to our method.

7) Solver: We use the Matlab (Mathworks, Inc.) function *fmincon* to solve the constrained gradient-based optimization problem. The (x, y) locations of the spline control points are confined by the bounds $(-10, 110)$; the derivatives of the control points are confined by the bounds $(-1000, 1000)$.

IV. Results

We present planned paths through four randomly generated obstacles fields. For all obstacle fields, $n_{\text{obs}} = 50$. To repeatedly simulate specific obstacle fields, the random number generator in Matlab was seeded with 1, 2, 3, and 4. Consequently, the obstacle fields will be referred to as rng(1), rng(2), rng(3), and rng(4) respectively. Table I reports the number of function evaluations required by each path (as reported by *fmincon*), the number of required path segments, and the total path length.

Figure 1 displays the evolution of the path in the rng(2) obstacle field for the multi-start approach; Figure 2 displays the resulting path for the single-start approach in the same obstacle field. Videos

TABLE I: Path Planning Results

Planning Scenario	Func. Eval.	Segments	Path Length
Single-Start, rng(2)	N/A	N/A	N/A
Single-Start, rng(4)	N/A	N/A	N/A
Multi-Start, rng(1)	10205	11	157.7259
Multi-Start, rng(2)	11275	11	153.6096
Multi-Start, rng(3)	11256	11	150.8103
Multi-Start, rng(4)	6849	11	145.0263

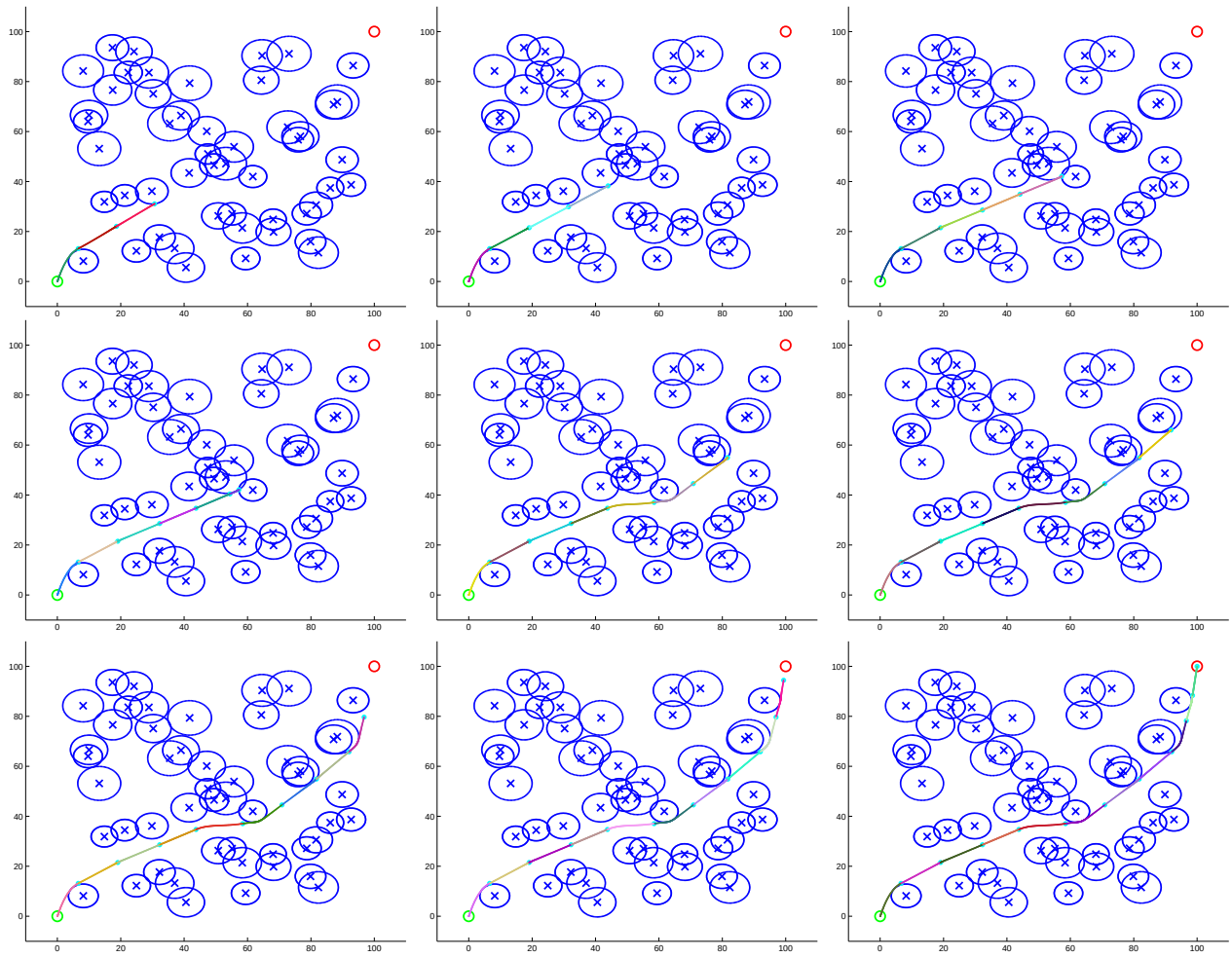


Fig. 1: The path produced using the multi-start approach in the rng(2) obstacle field. A viable path was found with little difficulty, even with groups of obstacles forming concavities.

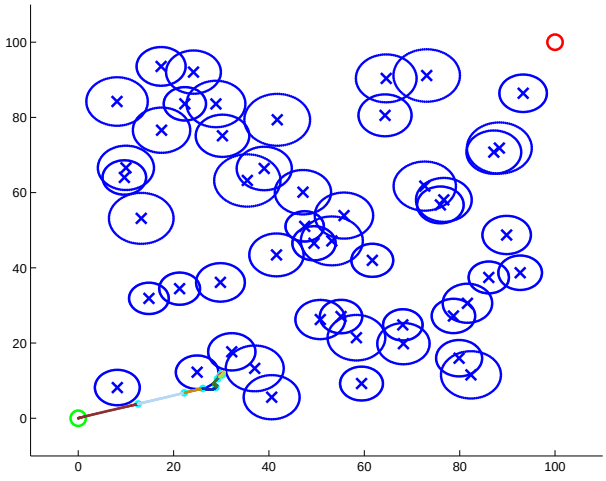


Fig. 2: The path produced using the single-start approach in the rng(2) obstacle field. The algorithm failed to find a viable path. As mentioned in section I-A, a drawback of receding horizon planners is their tendency to become stuck in obstacles with concave shape, as shown here.

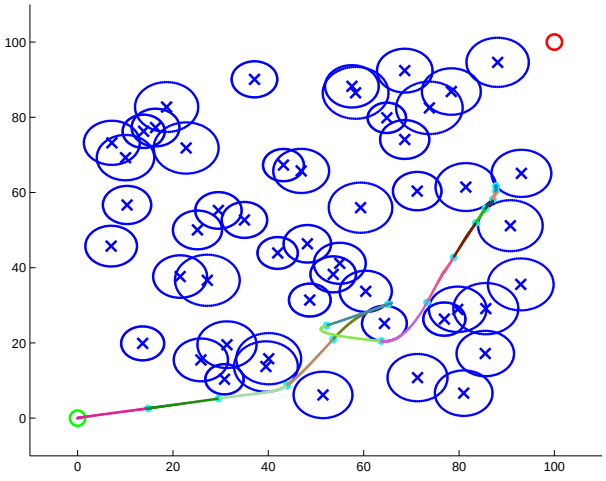


Fig. 3: The path produced using the single-start approach in the rng(4) obstacle field. The planner failed to find a viable path.

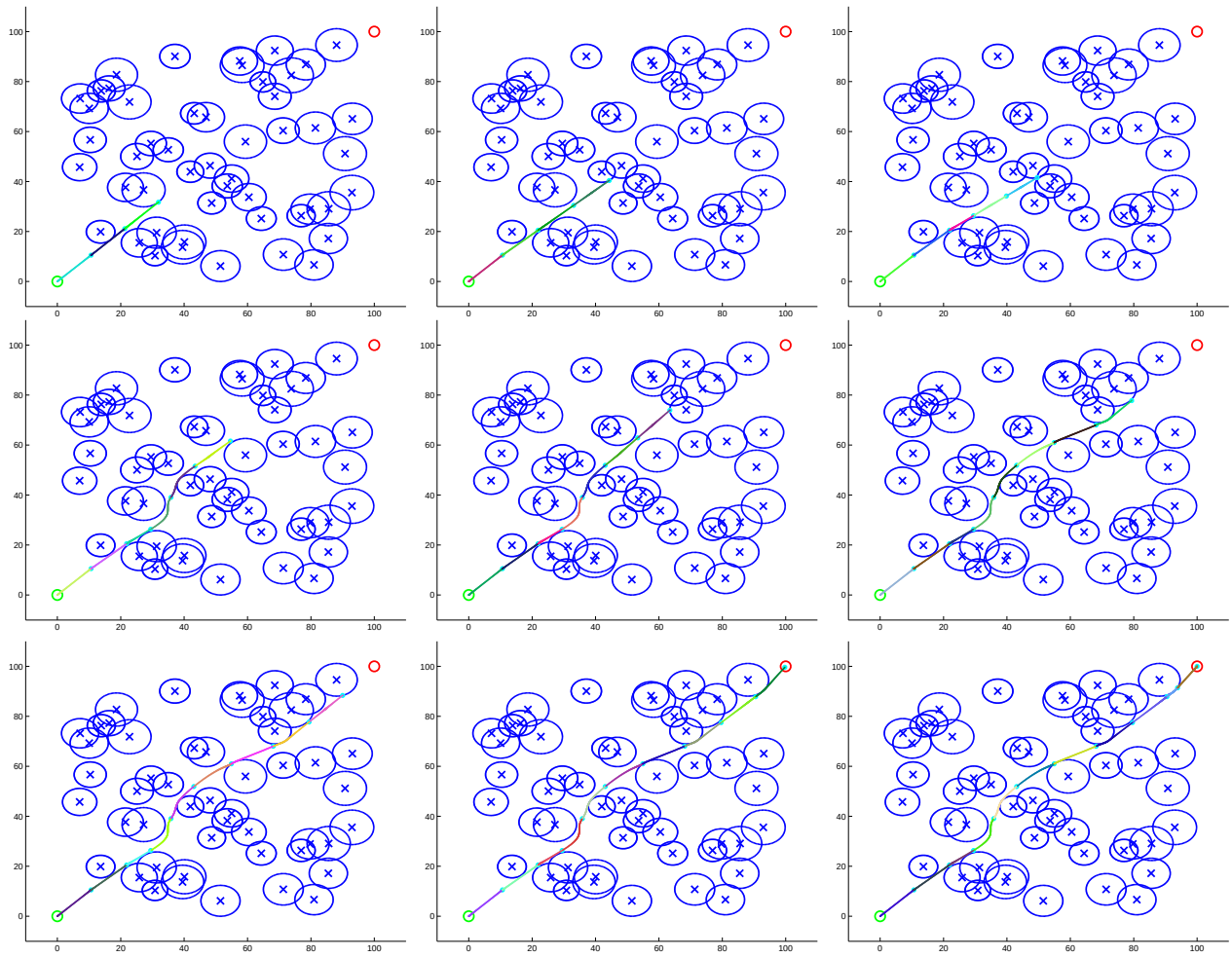


Fig. 4: The path produced using the multi-start approach in the rng(4) obstacle field. A viable and very efficient path was found through this challenging obstacle field.

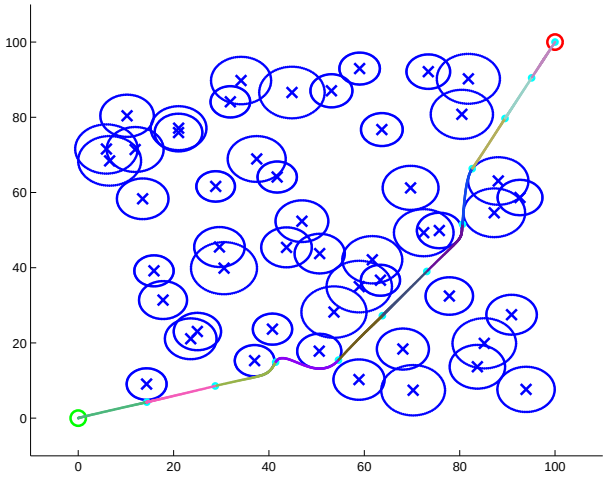


Fig. 5: The path produced using the multi-start approach in the rng(1) obstacle field. The multi-start approach allowed the planner to avoid a potentially dangerous group of obstacles.

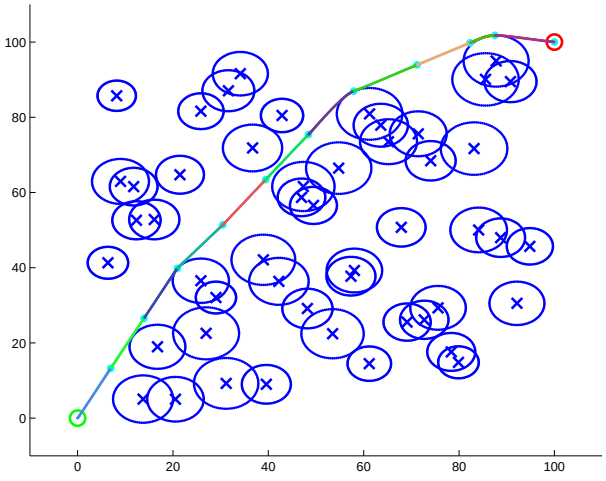


Fig. 6: The path produced using the multi-start approach in the rng(3) obstacle field. The planner easily found a path through this obstacle field.

showing the evolution of these paths are located at https://www.youtube.com/watch?v=XJn0Dau1Q84 (single-start) and https://www.youtube.com/watch?v=wiOkw5A-8Gw (multi-start). In this obstacle field, only the multi-start approach found a viable path to the final destination. We note that in the fourth panel of Figure 1, the path is temporarily trapped in an obstacle concavity—a local minimum of our objective function. The multi-start approach enables the path to escape this concavity and continue towards the final destination. Our objective function does possess some inherent robustness to concavities that often trap the single-start method.

Figure 3 displays the evolution of the path in the rng(4) obstacle field for the single-start approach; Figure 4 displays the resulting path for the multi-start approach in the same obstacle field. Videos showing the evolution of these paths can be found at https://www.youtube.com/watch?v=1h9zIFSA0xk (single-start) and https://www.youtube.com/watch?v=9LRA9OPhM5g (multi-start). This obstacle field is particularly difficult because of the barrier of contiguous obstacles centered at about $x = 50, y = 40$. The multi-start approach is trapped behind this barrier for one planning period, but is then able to proceed towards the final destination. The multi-start approach requires relatively few planning periods and plans a path that is only slightly longer than the obstacle-free distance to the final destination. In the end, the single-start approach is unable to find a viable path; it finds a way to circumvent the aforementioned barrier—though in a way not likely compatible with fixed-wing UAV dynamics—but then is still trapped later by other obstacles.

Figure 5 shows the path planned by the multi-start approach through the rng(1) obstacle field. Notice in this figure that the third and fourth path segments (olive green and purple) begin traveling upwards before proceeding to the right and then upwards. At these planning steps, the future path segments actually extended upwards into the open area directly above the fourth path segment. The multi-start approach allowed the path to re-route away from this dangerous area (containing a deep concavity). It does not necessarily do so intelligently, the optimizer always seeks only to reduce the final objective function at each step. The multi-start approach simply has the advantage of having more opportunities to find the globally optimum path at each step.

Increasing the $P_{\ell_{\max}}$ parameter effectively permits the optimizer to see farther at each planning step and can help reduce path length by minimizing the number of these types of directional changes. However, increasing $P_{\ell_{\max}}$ also expands the design space, thus increasing computational time.

Figure 6 shows the path planned by the multi-start approach through the rng(3) obstacle field. In this randomly-generated obstacle field there happens to be a relatively obstacle-free channel leading in the general direction of the final destination, which greatly facilitated the path planning.

V. Concluding Remarks

There are a number of straightforward extensions to this path planning method. These include considering non-circular obstacles, modeling vehicle dynamics, and extending to 3D space. Implementing obstacles of different shapes would not be difficult, it would simply require redefinition of the constraints, though it would increase the complexity of the problem. It may also just be just as efficient to model complex shapes with overlapping circles.

Another extension of the path planning algorithm is a more thorough investigation of how constraints on vehicle dynamics can be applied to constraints on spline parameters. Though we did a cursory exploration of this area, we did not have success in applying motion models to the spline shapes. Velocity dynamics could be included in this model: since splines use parametric representations, they already implicitly represent vehicle velocity. In this case, the optimization could be run for fastest path, not just the shortest path. If our algorithm were implemented in hardware, the curvature of the path would be sampled at each point in time and these curvature values could be used to inform ground speed commands to the UAV's flight controller.

In addition, the method here only considers searches in a 2D space. The search could be extended to 3D space, though some extension of 2D Ferguson splines into 3D space would need to be considered.

There are also more extensive changes that could be made to develop this path planner into a functional method. The first is determining if there is a reformulation of the problem so that it has a

deterministic computation time. For example, the methods listed in [6], [7] often have a proven big-O computational complexity. With the constrained optimization approach used here, it is difficult to guarantee when the algorithm will converge. It would be worth reconsidering the problem to see if our receding horizon approach could be presented as, for example, a convex optimization problem, which has very predictable behavior. At the very least, analytic or sensitivity-based gradients could be used rather than complex step to reduce computation time.

A second extensive change would be to determine how the receding horizon approach could be more robustly implemented. If the optimizer were to get truly stuck, perhaps in a wide, deep concavity, some method of backtracking and picking a new route would be necessary. One possibility is to keep a map of the space and update regions as visited as the path goes through them. If the planner gets stuck, it could then pick a favorable not-previously-visited region of the map, backtrack to a branch point, and proceed into the new region.

In pursuing these research directions, it will be important to do a more thorough comparison to other UAV path planners. There are many other approaches to path planning that are being studied and published. Though we performed a cursory literary search on the topic, the wide variety of methods available warrants a broader search through the discipline. A complete investigation would include performance benchmark comparisons to other efficient path planners.

REFERENCES

[1] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft*. Princeton University Press, Princeton, New Jersey, 2012.

[2] L.E. Kavraki, L.E. Kavraki, P. Svestka, P. Svestka, J.-C. Latombe, J.-C. Latombe, M.H. Overmars, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566 – 580, 1996.

[3] Po Pettersson and Patrick Doherty. Probabilistic roadmap based path planning for an autonomous unmanned helicopter. *Journal of Intelligent and Fuzzy Systems*, 17:395–405, 2006.

[4] Fei Yan, Yan Zhuang, and Jizhong Xiao. 3D PRM based real-time path planning for UAV in complex environment. In *2012 IEEE International Conference on Robotics and Biomimetics, ROBIO 2012 - Conference Digest*, pages 1135–1140, 2012.

[5] Myungsoo Jun and Raffaello D Andrea. Path Planning for Unmanned Aerial Vehicles in Uncertain and Adversarial Environment. *Cooperative Control: Models, Applications and Algorithms*, pages 95–111, 2003.

[6] C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 57(1-4):65–100, 2010.

[7] Qi Juntong Liang Yang, Jizhong Xiao, and Xia Yong. A Literature Review of UAV 3D Path Planning. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 2376–2381, 2014.

[8] J. Bellingham, A. Richards, and J.P. How. Receding horizon control of autonomous aerial vehicles. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 5, pages 3741–3746 vol.5. IEEE, 2002.

[9] Martin Saska, Martin Macas, Libor Preucil, and Lenka Lhotska. Robot Path Planning using Particle Swarm Optimization of Ferguson Splines. In *2006 IEEE Conference on Emerging Technologies and Factory Automation*, pages 833–839. IEEE, September 2006.

[10] E Mansury. Artificial bee colony optimization of Ferguson splines for soccer robot path planning. In *Proceeding of the 2013 RSI International Conference on Robotics and Mechatronics*, volume 1, pages 85–89, 2013.

[11] Elahe Mansury, Alireza Nikookar, and Mostafa E. Salehi. Differential evolution optimization of ferguson splines for soccer robot path planning. In Ali Movaghar, Mansour Jamzad, and Hossein Asadi, editors, *Artificial Intelligence and Signal Processing*, volume 427 of *Communications in Computer and Information Science*, pages 311–319. Springer International Publishing, 2014.

[12] J. Ye and R. Qu. Fairing of parametric cubic splines. *Mathematical and Computer Modelling*, 30(5-6):121–131, 1999.

[13] Joseph Scott Holub. Improving particle swarm optimization path planning through inclusion of flight mechanics. Master's thesis, Iowa State University, 2010.

[14] Joaquim Martins. *Multidisciplinary Design Optimization*. Unpublished, 2015.