Sequential Convex Programming for Nonlinear Optimal Control in UAV Trajectory Planning

Yong Li *, Qidan Zhu * and Ahsan Elahi

College of Intelligent Systems, Science and Engineering, Harbin Engineering University, Harbin 150001, China; ahsan.elahi@hrbeu.edu.cn
* Correspondence: llyy5927@hrbeu.edu.cn (Y.L.); zhuqidan@hrbeu.edu.cn (Q.Z.); Tel.: +188-4516-3353 (Y.L.)

Abstract: In this paper, an algorithm is proposed to solve the non-convex optimization problem using sequential convex programming. An approximation method was used to solve the collision avoidance constraint. An iterative approach was utilized to estimate the non-convex constraints, replacing them with their linear approximations. Through the simulation, we can see that this method allows for quadcopters to take off from a given initial position and fly to the desired final position within a specified flight time. It is guaranteed that the quadcopters will not collide with each other in different scenarios.

Keywords: trajectory planning; sequential convex programming; quadcopter; equality constraints; inequality constraints

1. Introduction

Quadcopters are the subject of ongoing research due to their agility, mechanical simplicity, and robustness. They are capable of performing a wide range of tasks and have been utilized as platforms for investigating vision-based pose estimation [1], nonlinear control [2], and learning [3], with many other applications. Moreover, they are helpful tools for solving practical issues, such as surveillance [4] and inspection [5]. Trajectory/path planning has obtained more attention as the interest in quadcopters has increased. These machines may interact with each other and the environment. As described in [6–8], recent trajectory optimization methods utilize a simple kinematics model and exclusively consider the acceleration vector as the decision variable. Consequently, the resulting trajectories only include information about velocities and positions. These methods assume the presence of a reliable onboard trajectory tracking controller, which justifies using a simple kinematics model. In other words, the trajectory planning algorithm independently controls each motor on the quadrotor to achieve stability. The early works [9,10] introduced traditional control methods for quadrotors, specifically a nonlinear geometric tracking controller on SE (3). These controllers can be classified as PID (proportional–integral–derivative) since they directly control the error dynamics in the attitude and position between the quadrotor and its desired target trajectory. After introducing traditional control methods, trajectory planning techniques have emerged as a distinct optimization issue, generally ignoring the actual dynamics of the drone.

In robotics, the generation of trajectories that adhere to avoidance constraints has been significantly studied. As stated in [11], there are two distinct approaches: planning and reacting. The planned strategy involves generating feasible courses in advance, whereas the reactive approach usually relies on an online collision avoidance system to respond to dangerous situations, as mentioned in [12]. Additionally, the two methods can also be combined. This work focused on a planned methodology. The planning can be performed in a decentralized manner [13] or in a centralized manner [14], as described in this paper. This study was conducted on trajectory generation with avoidance constraints for numerous vehicles, including differential drive robots [15], underwater vehicles [16],...
and aircraft [17]. Pattern formation research [18,19] also relies on collision-free trajectory generation. Various convex optimization techniques are employed to find collision-free trajectories while considering specific performance criteria. A model predictive control problem is described in [20], which examined highly specialized quadcopter dynamics and subsequently derived the constraints. The generation of trajectories involves constructing the trajectory of the quadcopter based on its jerk and solving a convex optimization problem for each independent axis. A technique for generating trajectories that accounts for simple dynamics constraints was proposed in [6], which is independent of the type of vehicle.

Various trajectory generation strategies have been proposed, depending on the desired objective. Advanced techniques for generating trajectories for quadcopters that consider the agent constraints can be found in [17,21]. This work aims to minimize the total thrust needed to fly the trajectories while using avoidance constraints to synchronize the trajectories of several vehicles. We assume that the agent can follow the generated trajectories (using an appropriate controller) if they satisfy a feasibility check. Separating the lengthy feasibility check from the trajectory planning algorithm results in a simple and flexible approach for rapidly generating collision-free trajectories for many vehicles. A method for planning aircraft trajectories that considers avoidance constraints and utilizes simplified dynamics was proposed in [10]. Two recent reviews [22,23] provide diverse research on motion planning for Unmanned Aerial Vehicles (UAVs). The complexity of trajectory generation is exacerbated by the under-actuated and nonlinear dynamics of the quadcopter and the challenging task of modeling aerodynamics (see [24,25] for studies on aerodynamic effects and [3] for a learning-based compensation technique). In [26], collision-free trajectories are computed to guide a fleet of UAVs from their starting positions to their destination positions, ensuring that the trajectories maintain a minimum separation distance while minimizing the total thrust produced by the quadcopters. The rapid solutions are obtained using sequential convex programming, with times in the range of seconds. In [27], the paper presents a convex programming approach for a fuel-optimal powered landing problem in the presence of aerodynamic drag forces and new types of non-convex control constraints.

Based on the classical methods proposed in the above literature, the mature business software [28] solves the linear problem quickly and effectively. Other methods treat the optimal control problem as a general nonlinear programming problem to better describe the actual model. However, in practice, it is of great difficulty to obtain a solution to nonlinear programming because of its nature, unbounded operation time, and the requirement for an initial guess, which should be provided by humans. Furthermore, the solution has the chance to be a local optimal rather than a global. Due to the above reasons, treating the optimal control problem as a general nonlinear programming approach for solving is not appropriate for onboard applications. To solve nonlinear programming stably and quickly is the primary goal of this paper. Conversely, the problem is solved perfectly if the nonlinear programming can be transformed into convex programming without losing the optimal solution. Here, the local optimal and global optimal are identical, and the solution time is bounded, therefore an initial prediction is unnecessary. Unfortunately, the transformation cannot be identical, and the optimality or the equivalence must be removed. The solution to the approximated problem, which identifies a key point of the original problem, is a feasible alternative. So, we need to find a convex programming formulation or a series of convex programming formulations whose solutions correspond to key points of the original problem. Approximation methodology plays a key role in the process. Regarding the above issue, this paper proposes an algorithm that iteratively uses convex optimization to approximate non-convex optimization.

The trajectory planning problem we considered involves solving the trajectory tracking problem using optimization techniques. There are various existing categories of optimization algorithms, such as convex optimization, non-convex optimization, etc. The primary objective of this work was to design and implement a sequential convex programming algorithm (SCP) for quadrotor trajectory planning. Sequential convex programming can be categorized as a non-convex optimization algorithm. Such methods rely on constructing
convex sub-problems via successive linearizations of the original (nonlinear) objective functions and constraints. The resulting sub-problems become convex optimization sub-problems, which are typically solved by existing convex programming solvers. Sequential convex programming (SCP) is a technique employed to plan trajectories for numerous agents in three-dimensional (3D) space [29]. The objective is to transition from an initial set of states, including each vehicle’s position, velocity, and acceleration, to a final set of states. This must be done while maintaining a minimum distance between the agents and satisfying additional trajectory constraints. The methodology is implemented on quadcopters (Figure 1) but can be easily adapted for various platforms by adjusting the constraints. This study was inspired by the research conducted on sequential convex programming (SCP) by Wang et al. [30].

![Figure 1. Dynamic model of a quadcopter.](image)

In this paper, we explored the capability of sequential convex programming and applied it to the quadrotor dynamics model, including trajectory dynamics. An iterative approach was used to estimate the non-convex constraints, replacing them with their linear approximations, and showing the simulation results. The simulation demonstrated that the approach can generate collision-free trajectories in different scenarios. The results demonstrate that this method can be used in real-world applications.

This study is outlined in the following order: In Section 2, trajectory dynamics are constructed. In Section 3, the constraints are explained in detail, including formulation of the problem as a convex optimization problem and discussion on using the iterative approach for approximating non-convex constraints. In Section 4, the simulation results are discussed. Section 5 contains the conclusion and an analysis of this paper.

2. Dynamic Model

Based on the given set of initial and final positions for N drones, the objective was to ensure that these drones reached their intended positions within a specified time interval, T, while keeping a minimum distance of at least R from each other during their trajectories. An optimization problem was formulated to accomplish this goal, which involved several convex and affine constraints and a non-convex collision avoidance constraint. The solution to this optimization problem guaranteed the achievement of the specified objective.
The goal was to generate trajectories for N agents that do not collide, allowing a transition from an initial set of states to a final set of states (position, velocity, and acceleration) within a specified time T. The trajectories must adhere to many boundaries and avoidance constraints.

The quadcopter was modeled as a rigid body with six degrees of freedom: it can linearly translate along the inertial x3 axes and rotate around these three inertial axes. The rotation is described by the frame attached to the body relative to the inertial frame, using the proper orthogonal matrix R. The control inputs to the system were taken as the total thrust produced, denoted as f, for simplicity, normalized by the vehicle mass, with units of acceleration. The body rates, expressed as $\omega = (\omega_1, \omega_2, \omega_3)$, are considered in the body-fixed frame. $(x_1, x_2, x_3)$ represents the position of the quadcopter in inertial space. An illustration of this model can be seen in Figure 1.

Euler angles describe the orientation of a rigid body in space. Each angle represents the motion, as follows:

$$\phi, \theta, \psi$$

where $\phi, \theta, \psi$ are the quadrotor’s roll, pitch, and yaw angles, respectively, s represents sin, and c represents cos.

The rotational dynamics of a quadcopter can be described by Euler’s equations of motion, as follows:

$$I \dot{\omega} + \omega \times (I \omega) = \tau$$

where I is the inertia matrix, $\omega$ is the angular velocity vector, and $\tau$ represents the torque vector.

The differential equations governing the flight of the quadcopter are now those of a rigid body, as follows:

$$\dot{x} = R_{x3} f + g$$

$$\dot{R} = R [ \omega \times ]$$

With $e_3 = (0, 0, 1)$ and $[ \omega \times ]$ denoting the skew-symmetric matrix form of the vector cross product, such that:

$$[ \omega \times ] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

and $g = (0, 0, -g)$ represents the acceleration due to gravity.

2.1. Trajectory Dynamics

Let $p_i^t$, $V_i^t$, and $a_i^t$ represent the location, velocity, and acceleration, respectively, of agent ‘i’ at discrete times $t$. The trajectory of agent ‘i’ is determined by the following discretized dynamics equations:

$$p_i^{t+1} = p_i^t + \Delta t V_i^t + \frac{\Delta t^2}{2} a_i^t$$

$$V_i^{t+1} = V_i^t + \Delta t a_i^t$$

where $\Delta t$ represents the duration of each time step. The variable $t$ represents the current time step, ranging from 0 to $T$. The variable $i$ represents the agent number from 1 to $N$. $T$ and $N$ represent the final time step and the total number of agents, respectively. The position and the velocity of each agent can be expressed as affine functions of acceleration.
as indicated by Equations (6) and (7). Equations (6) and (7) can be iteratively expanded to be written explicitly as a function of the accelerations of the vehicle at the previous time steps. Therefore, the velocity of quadcopter \( i \) at time \( t \) is determined as follows:

\[
V^t_i = V^0_i + \Delta t \sum_{k=1}^{t} a^k_i
\]  
(8)

From Equations (6) to (8), the position of quadcopter \( i \) at time \( t \) is represented by the following:

\[
P^t_i = P^0_i + \Delta t \frac{V^0_i}{2} \sum_{k=1}^{t} a^k_i \left(2t - 2k + 1\right)
\]  
(9)

where \( t \) ranges from 1 to \( T \), and \( i \) ranges from 1 to \( N \). The \( V^0_i \) is equal to 0, so Equations (8) and (9) can be simplified as follows:

\[
V^t_i = \Delta t \sum_{k=1}^{t} a^k_i
\]  
(10)

\[
P^t_i = P^0_i + \frac{\Delta t^2}{2} \sum_{k=1}^{t} a^k_i \left(2t - 2k + 1\right)
\]  
(11)

Equations (10) and (11) can be utilized to create an optimization problem involving the optimization parameter \( X \). This parameter represents the acceleration of each agent at every time step, i.e., \( X \in \mathbb{R}^{3NT} \).

2.2. Objective Function

The control inputs for a quadcopter consist of the total thrust produced and the angular velocities. This study examined a normalized thrust force expressed in units of acceleration for the sake of simplicity.

\[
a = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}
\]  
(12)

where \( R \) represents the rotation matrix that accurately describes the rotation between the inertial and body-fixed frames.

The quadcopter is described by six degrees of freedom. The translational position \((x_1, x_2, x_3)\) is measured in the inertial coordinate system, as shown in Figure 1. The quadcopter attitude \( b \) is defined by the rotation matrix \( R \). The rotation matrix is defined such that when multiplying a vector \( v \) in the quadcopter coordinate \( b \) system with it, the same vector, described in the inertial coordinate system \( i \), is obtained.

\[
i v = b R^t i v
\]  
(13)

The input thrust \( f \) can be calculated by applying the Euclidean norm to Equation (12), as follows:

\[
f = a - g
\]  
(14)

where \( g \) is the acceleration due to gravity and is equal to \((0, 0, -g)\) in its vector form. To determine the jerk, it is necessary to express \( \dot{R} \) in terms of the known parameters. The expression of \( \dot{R} \) can be obtained by using the rotation matrix, \( R \), and the skew-symmetric matrix of body rates, as follows:

\[
\dot{R} = R \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}
\]  
(15)
By applying Equations (13) and (15) and taking the derivative of the acceleration, the jerk can be calculated as follows:

\[ j = \dot{\mathbf{R}} \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} + \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \]  

(16)

\[ j = \mathbf{R} \left( \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ -\omega_3 & 0 & -\omega_1 \\ \omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} \right) \]  

(17)

Therefore, the cumulative thrust exerted by all agents at every time step can be selected as the cost function, as follows:

\[ f_0 = \sum_{t=1}^{T} \sum_{i=1}^{N} a_t^i + g^2 \]  

(18)

Equation (16) can be formulated as a quadratic equation, with \( G \) representing the gravity vector of each agent at every time step, i.e., \( G \in \mathbb{R}^{3NT} \), as follows:

\[ f_0 = \mathbf{X}^T \mathbf{X} + 2\mathbf{G}^T \mathbf{X} + \mathbf{G}^T \mathbf{G} \]  

(19)

Equation (18) is expressed in a quadratic form. Hence, employing this cost function alongside appropriate affine equality/inequality constraints is feasible. Therefore, this issue can quickly be resolved using existing quadratic programming solvers.

Once the acceleration of the agents is determined at each time step, the total thrust force may be calculated using Equation (14). By rearranging Equation (16), we may derive the following equation:

\[ \begin{bmatrix} \omega_2 \\ -\omega_1 \\ 0 \end{bmatrix} = \frac{1}{f} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}^T j \]  

(20)

The absence of the third component of body rates, \( \omega_3 \), can be explained by observing that a rotation about the \( \omega_3 \) axis does not impact the translational acceleration. The system inputs for a trajectory defined in its jerk are given using Equations (14) and (20), with one degree of freedom left in \( \omega_3 \). This could be fully specified if the total attitude of the quadcopter, including rotation around the thrust axis, is also known. For simplicity, we will assume that the value of \( \omega_3 \) is zero and that this rotation is insignificant.

In essence, the output of the optimization problem, which is the acceleration of the agents at each time step, may be used to calculate suitable inputs, body rates, and thrust quickly.

3. Problem Formulation

3.1. Convex Constraints

In this section, we describe the trajectory planning problem as an optimization problem. Therefore, we define an objective function that we aim to minimize, a variable that may be optimized freely, simple dynamics equations that describe the evolution of the trajectories, and constraints that must be met regarding inequalities and equalities. The proposed method can generate trajectories for vehicles of any arbitrary nature.

Imposing several constraints while minimizing accelerations is necessary to achieve the desired objective. The initial and final positions of all the vehicles are known. Additionally, to accommodate the physical constraints of an agent, velocity, acceleration, and jerk may be restricted. Finally, we need to ensure that the generated trajectories maintain a minimum distance of \( R \) between each other at all time steps, thereby confirming collision avoidance.

Next, we will establish and express these constraints in matrix form. To simplify the representation, we analyzed one vehicle at a time and then combined these constraints to form the constraints for all the agents.
3.2. Equality Constraints

To begin the algorithm, it is necessary to select the initial velocity and the position parameters. Furthermore, they must decelerate when they near their individual goals until they reach complete rest. Therefore, their final velocities and accelerations must also be zero. Equations (6) and (7) can be utilized to enforce the constraints on initial velocity and position. Equality constraints can enforce final acceleration, velocity, and position constraints. Considering that the constraints apply to each axis for final acceleration, velocity, and position, there must be nine equality constraints in total for every agent. Therefore, it is necessary to have a total of $9N$ boundary constraints in the system. These constraints can be expressed as equality constraints in the following manner:

Final velocity constraint: $v^f_i = 0_{3 \times 1}$
Final acceleration constraint: $a^f_i = 0_{3 \times 1}$
Final position constraint: the final positions in the form of $x_f$. For a single agent $i$, we have the following:

$$p^f_i = x_{f,i}$$ (21)

The matrix representation of these constraints can be written as follows:

$$A_{eq,f,p,i} \chi_i = b_{eq,f,p,i}$$ (22)

After establishing the equality constraint matrices for a single agent, our next step was to combine these constraints into a unified matrix. The equality constraints for a single vehicle, $i$, can be stacked on top of each other, resulting in a matrix representation of the equality constraints as follows:

$$A_{eq,i} \chi_i = b_{eq,i}$$ (23)

We have now created the matrices representing the equality constraints for a single vehicle. We generalized this concept to encompass the matrices that express all the equality constraints for all the agents.

$$A_{eq} X = b_{eq}$$ (24)

where $A_{eq} \in \mathbb{R}^{9N \times 3N(T+1)}$ and $b_{eq} \in \mathbb{R}^{9N}$.

3.3. Inequality Constraints

Position constraint: The motion range in the workspace is physically limited by maximum position ($p_{\text{max}}$) and minimum position ($p_{\text{min}}$). Thus, given a single agent $i$ at time step $k (>1)$, we can write the position constraints as follows:

$$p_{\text{min}} \leq p^f_i \leq p_{\text{max}}$$ (25)

Velocity constraint: To ensure safe flight in a confined space and dynamic feasibility, the velocity of the UAVs must be limited by minimum velocity ($v_{\text{min}}$) and maximum velocity ($v_{\text{max}}$). Therefore, for a single agent $i$ at time step $k (>1)$, we can express velocity constraints as follows:

$$v_{\text{min}} \leq v^f_i \leq v_{\text{max}}$$ (26)

Acceleration constraint: To ensure a smooth and dynamically feasible trajectory, it is necessary to limit the acceleration of all the vehicles at all times within the range defined by minimum acceleration ($a_{\text{min}}$) and maximum acceleration ($a_{\text{max}}$). Thus, for a specific vehicle $i$ at time step $k (>1)$, the acceleration constraints can be expressed as follows:

$$a_{\text{min}} \leq a^f_i \leq a_{\text{max}}$$ (27)

Jerk constraint: Limiting a specific vehicle’s jerk ($j_i$) is essential. Jerk can be defined as the first derivative of the acceleration to maintain smooth acceleration in the quadcopter [2,3]. Jerk must be bounded for smooth and feasible trajectories. For simplicity, the jerk can be constrained in each axis individually.

$$j_{\text{min}} \leq j^f_i \leq j_{\text{max}}$$ (28)
The equation below expresses the jerk of any agent at any time \( j^t_i \) in discrete time.

\[
j^t_i = \frac{a^t_i - a^{t-1}_i}{\Delta t}
\]

(29)

By combining all these constraints, we derive the following inequality:

\[
A_{\text{feasibility}} X \leq b_{\text{feasibility}}
\]

(30)

where \( A_{\text{feasibility}} \in \mathbb{R}^{6N(2T+1) \times 3N(T+1)} \) and \( b_{\text{feasibility}} \in \mathbb{R}^{6N(2T+1)} \).

The inequality constraints established so far constitute a convex feasible set. However, the collision avoidance constraint is formulated as a non-convex constraint. We employed the SCP algorithm to obtain an affine approximation of this constraint. Subsequently, we constructed the total inequality matrices for the true optimization variable \( X \).

3.4. Collision Avoidance Constraint

To prevent the drones from colliding, it is necessary to maintain a minimum distance between them at each time step.

\[
\|p^t_i - p^t_j\|_2 \geq R \quad \forall i, j \neq j, \forall t
\]

(31)

where \( R \) is the collision threshold distance. Equation (6) is linearized as follows:

\[
p^t_{i,q} - p^t_{j,q} + \left(\frac{p^t_{i,q} - p^t_{j,q}}{p^t_{i,q} - p^t_{j,q}}\right)^T (p^t_i - p^t_j) \geq R
\]

(32)

where the superscript \( p^t_{i,q} \) denotes the position vector of agent ‘i’ at time step ‘k’ in the previous outer-loop iteration ‘q’.

The Collision Avoidance Constraint can be derived as follows:

\[
A_{\text{collision}} X \leq b_{\text{collision}}
\]

(33)

By combining Equations (28) and (31), we obtain the following inequality:

\[
AX \leq b
\]

(34)

An optimization problem can be solved in two phases. In the first step, we disregard the collision avoidance constraint and construct our quadratic programming (QP) problem, relying on the position, velocity, acceleration, and jerk constraints. Consequently, a single agent’s total inequality coefficient and residual matrices are obtained by vertically stacking all the coefficients and residual matrices. The result \( \hat{X} \) is subsequently employed to commence the second phase. Starting from the second iteration, we can apply the collision avoidance constraints because we now have a previous solution that serves as a reference to simplify the collision avoidance constraint.

1. We found an initial value by using the feasibility constraints.

\[
\min_{X} f_0
\]

which is subjected to the following:

\[
A_{eq} X = b_{eq}
\]

\[
A_{\text{feasibility}} X \leq b_{\text{feasibility}}
\]

which will determine \( \hat{X} \).

2. We utilized the estimated value \( \hat{X} \) to create new inequality matrices \( A \) and \( b \).
(3) \[ \min f_0 \]

which is subjected to \[ A_{eq}X = b_{eq} \]

\[ AX \leq b \]

During the second step, an iterative approach is used, where the most recent estimation of \( X \) \( \hat{x} \) is applied in each iteration, assuming that \( f_{\text{threshold}} \) is a specified threshold. Step two is repeated if \( f_0 - (f_0)_{\text{prev}} > = f_{\text{threshold}} \).

Because all the constraints are either convex or convexified, quadratic programming tools can be used to solve the optimization problem. The following is the pseudocode of this algorithm. \( A_{1,\text{feasibility}} \) represents the value in the first iteration of \( A_{\text{feasibility}} \) and \( b_{1,\text{feasibility}} \) represents the value in the first iteration of \( b_{\text{feasibility}} \).

3.5. Sequential Convex Programming

Sequential Convex Programming (SCP) is a local optimization method for non-convex problems that leverage convex optimization.

We considered the non-convex problem as follows:

\[ \min f_0(x) \]

\[ s.t. f_i(x) \leq 0, \forall i = 1, 2, \ldots, m \]

\[ h_j(x) = 0, \forall j = 1, 2, \ldots, p \]

with \( x \in \mathbb{R}^n \), \( f_0(x) \) non-convex and \( h_j \) non-affine.

In the \( k^{th} \) iteration of SCP, we maintained the estimate \( x^{(k)} \) of the solution, and the convex trust region \( \tau^{(k)} \subset \mathbb{R}^n \). We constructed convex approximations \( \overline{f}_0 \) and \( \overline{f}_i \) of \( f_0 \) and \( f_i \), respectively, and affine approximations \( \overline{h}_j \) of \( h_j \) over the trust region \( \tau^{(k)} \). The approximate convex problem obtained is written as follows:

\[ \min \overline{f}_0(x) \]

\[ \overline{f}_i(x) \leq 0, \forall i = 1, 2, \ldots, m \]

\[ \overline{h}_j(x) = 0, \forall j = 1, 2, \ldots, p \]

The solution to this approximate problem is \( x^{(k+1)} \).

In summary, the specific algorithm for trajectory optimization using SCP is depicted as Algorithm 1:

<table>
<thead>
<tr>
<th>Algorithm 1: Trajectory Optimization Using SCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Require: N,R,T,f_{\text{threshold}},p_{\text{max}},p_{\text{min}},v_{\text{max}},v_{\text{min}},a_{\text{max}},a_{\text{min}},j_{\text{max}},j_{\text{min}}</td>
</tr>
<tr>
<td>1: ( x^0 \leftarrow \text{QP}(A_{eq},b_{eq},A_{1,\text{feasibility}},b_{1,\text{feasibility}}) )</td>
</tr>
<tr>
<td>2: ( f_p \leftarrow f(x^0) )</td>
</tr>
<tr>
<td>3: ( n_1 \leftarrow 0 )</td>
</tr>
<tr>
<td>4: while ( n_1 \leq m_1 ) do</td>
</tr>
<tr>
<td>5: ( X \leftarrow \text{QP}(A_{eq},b_{eq},A_{\text{feasibility}},b_{\text{feasibility}}) )</td>
</tr>
<tr>
<td>6: ( f_c \leftarrow f(X) )</td>
</tr>
<tr>
<td>7: if (</td>
</tr>
<tr>
<td>8: break</td>
</tr>
<tr>
<td>9: end if</td>
</tr>
<tr>
<td>10: ( f_p \leftarrow f_c )</td>
</tr>
<tr>
<td>11: ( n_1 \leftarrow n_1 + 1 )</td>
</tr>
<tr>
<td>12: end while</td>
</tr>
</tbody>
</table>

4. Experiment

This section presents four case studies and demonstrates the performance of the suggested strategy through simulation data. The algorithm was executed using MATLAB
R2022b on a PC with an Intel i9-13900HX CPU operating at a clock cycle of 2.2 GHz and 16.0 GB RAM. Specifically, we used MATLAB’s quadratic programming function, quadprop. The case study examines two, four, six, and eight quadcopters operating within a three-dimensional space of $5 \times 5 \times 3$, respectively. Initially, the simulation was executed without imposing any distance constraints. Figures 2a, 3a, 4a and 5 illustrate that the algorithm produced the optimal trajectories, i.e., the shortest distances between the initial and final locations. Due to the absence of constraints on collision, the trajectories intersect at some points.

For the first iteration of the optimization loop of SCP, we did not consider the collision avoidance constraint and we formulated our QP problem using only the position, velocity, acceleration, and jerk constraints; therefore, the total inequality coefficient matrices, as well as the residual matrices, for a single drone were obtained by stacking all the coefficient matrices and residual matrices vertically.

In the first iteration, we did not have any previous solution to linearize our collision constraint; therefore, we did not consider the collision constraint in the first iteration. However, from the second iteration onwards, we enforced the collision avoidance constraints, since we had a previous solution around which we could linearize the collision avoidance constraint.

Figures 2–10 show the simulation results of two, four, six, and eight quadcopters in the absence of collision avoidance constraints and collision-free.

The parameters used for the case study were chosen as follows:

(1) Number of agents = 2
Simulation length = 30 s
Discretization step-size, $dt = 0.2$ s
Cost Threshold, $f_{threshold} = 0.05$
Run time for the first step of the algorithm = 2.45 s
Total run time = 35 s

(2) Number of agents = 4
Simulation length = 30 s
Discretization step-size, $dt = 0.2$ s
Cost Threshold, $f_{threshold} = 0.05$
Run time for the first step of the algorithm = 4.86 s
Total run time = 67 s

(3) Number of agents = 6
Simulation length = 30 s
Discretization step-size, $dt = 0.2$ s
Cost Threshold, $f_{threshold} = 0.05$
Run time for the first step of the algorithm = 6.75 s
Total run time = 94 s

(4) Number of agents = 8
Simulation length = 30 s
Discretization step-size, $dt = 0.2$ s
Number of inequality constraints without distance constraints = 7200
Total number of inequality constraints with distance constraints = 11,400
Cost Threshold, $f_{threshold} = 0.05$
Number of iterations for convergence = 2
Run time for the first step of the algorithm = 10.85 s
Total run time = 168 s
Figure 2. (a) Trajectories for two quadcopters in the absence of collision avoidance constraints. (b) Collision-free trajectories for two quadcopters.
Figure 3. (a) Trajectories for four quadcopters in the absence of collision avoidance constraints. (b) Collision-free trajectories for four quadcopters.
Figure 4. (a) Trajectories for six quadcopters in the absence of collision avoidance constraints. (b) Collision-free trajectories for six quadcopters.
Figures 5 and 6 show the variation of position and velocity as time of eight quadcopters in the absence of collision avoidance constraints. Figures 9 and 10 show the variation of position and velocity as time of eight quadcopters with collision-free constraints.

**Figure 5.** Trajectories for eight quadcopters in the absence of collision avoidance constraints.

During the execution of the second stage of the algorithm, the trajectories were automatically adjusted by the algorithm to prevent collision. Figures 2b, 3b, 4b and 8a,b demonstrate that deviation in the trajectories was minimal while ensuring that the trajectories did not intersect or collide. In these specific case studies, the minimum distance was defined as 1 m ($d_{\text{min}} = 1$). Depending on the quadcopter’s size and the size of the environment, this parameter was chosen accordingly. The selection of this parameter can be adjusted based on the dimensions of the quadcopter and the surrounding environment. Figure 11 shows the objective value plotted against the number of iterations. It can be seen from Figure 10 that the objective function value rose suddenly after the first iteration and continued to decrease from the second iteration onwards. This behavior was expected since, for the first iteration, the optimization problem was solved without the collision constraint. Following the first iteration, the collision constraint was taken into account, which reduced the objective value and resulted in a collision-free optimal trajectory.
Run time for the first step of the algorithm = 10.85 s

Total run time = 168 s

Figures 5 and 6 show the variation of position and velocity as time of eight quadcopters in the absence of collision avoidance constraints. Figures 9 and 10 show the variation of position and velocity as time of eight quadcopters with collision-free constraints.

Figure 5. Trajectories for eight quadcopters in the absence of collision avoidance constraints.

Figure 6. Position vs. time of eight quadcopters in the absence of collision avoidance constraints.

During the execution of the second stage of the algorithm, the trajectories were automatically adjusted by the algorithm to prevent collision. Figures 2b, 3b, 4b, and 8a,b demonstrate that deviation in the trajectories was minimal while ensuring that the trajectories did not intersect or collide. In these specific case studies, the minimum distance was defined as 1 m ($d_{min} = 1$). Depending on the quadcopter's size and the size of the environment, this parameter was chosen accordingly. The selection of this parameter can be adjusted based on the dimensions of the quadcopter and the surrounding environment. Figure 11 shows the objective value plotted against the number of iterations. It can be seen from Figure 10 that the objective function value rose suddenly after the first iteration and continued to decrease from the second iteration onwards. This behavior was expected since, for the first iteration, the optimization problem was solved without the collision constraint. Following the first iteration, the collision constraint was taken into account, which reduced the objective value and resulted in a collision-free optimal trajectory.
Figure 8. (a) Collision-free trajectories for eight quadcopters. (b) Collision-free trajectories for eight quadcopters.
Figure 9. Position vs. time of eight quadcopters with collision-free constraints.

Figure 10. Velocity vs. time of eight quadcopters with collision-free constraints.
Figure 11. Objective value plotted against the number of iterations.

5. Conclusions

In this study, an algorithm is presented that enables the generation of collision-free trajectories for a multi-agent quadcopter fleet. The non-convex optimization problem was iteratively solved using sequential convex programming that approximates non-convex constraints by using convex constraints. Through the simulation, we can see that this method allows for quadcopters to take off from a given initial position and fly to the desired final position within a specified flight time. It is guaranteed that the quadcopters will not collide with each other. Since the control effort was minimized, these trajectories are the shortest possible trajectories under the collision avoidance constraints. The simulation results show that, in different scenarios, this method can generate collision-free trajectories from the beginning point to the target point. Furthermore, the simulation results demonstrate the computational effort, and the performance indicates that this algorithm can be used in real-life applications with well-tuned hyper-parameters. These collision-free trajectories can be generated online.

Author Contributions: Conceptualization, Q.Z. and Y.L.; methodology, Y.L. and Q.Z.; software, Y.L. and A.E.; validation, Y.L. and A.E.; formal analysis, Y.L.; investigation, Y.L.; resources, Q.Z.; writing-original draft preparation, Y.L.; writing-review and editing, Y.L. and A.E.; visualization, Y.L.; supervision, Q.Z.; project administration, Q.Z.; funding acquisition, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References


**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.