Multiagent UAV Trajectory Optimisation using Sequential Convex Programming

ME308
QuadKopters (Group 12)

Under the guidance of
Prof. Avinash Bhardwaj
And mentorship of
Mr. Gopalan Iyengar
Department of Mechanical Engineering

October 15, 2023

Contents

1 Introduction

Unmanned Aerial Vehicles (UAVs) find applications in multiple domains such as surveillance, mapping, cinematography, rescue missions, military, and many more. Various tech giants such as Amazon, and Walmart have recently adopted UAV delivery systems. To make a machine such as a quadcopter autonomously perform these complicated tasks, there are multiple disciplines that have to synergize to bring about, not only the completion of these tasks but also to ensure safety during the process.

- **Vision :** A well-equipped UAV requires sensors for detecting and tracking objects. An RGB image from an onboard camera is typically needed for most object detection and tracking tasks, while an RGBD or stereo camera's depth image input is necessary for applications like mapping and navigation.
- **Motion Planning :** The UAV needs to plan a collision-free and optimal trajectory to reach its goal based on the input from its onboard camera. This trajectory information typically includes the state variables' values (consisting of pose $\mathbf{x} = \left[x, y, z, \phi, \theta, \psi\right]^T$ and its derivatives), which the UAV must track to achieve the goal while ensuring a safe traversal. Trajectory planning is a crucial element in any autonomous mission.
- **Localization :**For autonomous missions, the UAV needs to understand its position relative to its body frame using a pose vector. State estimation is crucial to track the desired trajectory and minimize the error between the desired and current trajectory. Using onboard sensors, the UAV can improve its estimate of the current state by utilizing information from the literature on state estimation, which is a necessary component of autonomous navigation.
- **Controls :** The planning problem generates a computational vector of pose and its derivatives for the desired trajectory, but the quadcopter flies based on voltage values as control inputs to the motors. Therefore, a specific sequence of control inputs must be used to track the desired trajectory, which is determined using the control theory literature and is a crucial aspect of autonomy.
- **Hardware :** It goes without saying that the hardware of the UAV must be able to withstand various aerodynamic stresses and vibrations that occur during flight. Additionally, the design should be optimized to ensure that the frame of the UAV is lightweight, agile, and robust.

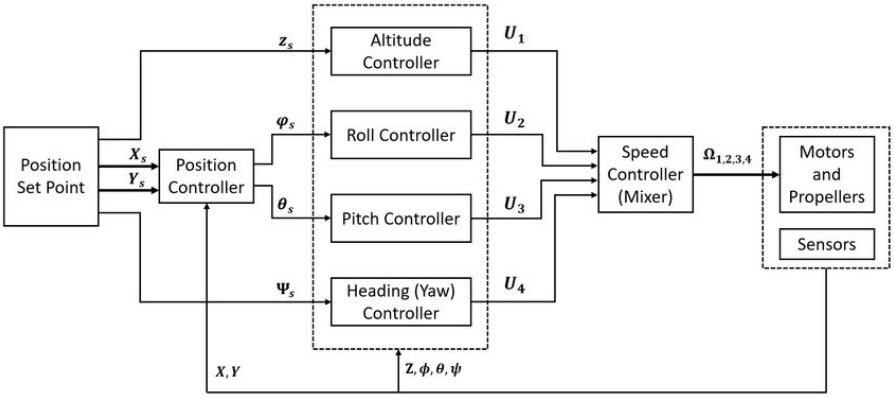


Figure 1: A Typical Flight Control System for a Quadcopter

1.1 Motivation

This report investigates the use of Sequential Convex Programming in applications such as multi-agent UAV optimal trajectory planning in 3D. Considering the idea of using UAVs for package delivery, having multiple drones in the same vicinity approaching their delivery location can increase the risk of collision between any two drones. In such a scenario when there are multiple aerial vehicles with distinct start and goal points, how can these vehicles collaboratively utilize the state information and plan optimal trajectories so as to ensure that they reach their goal point in desired time as well as navigate through a collision-free trajectory? Here optimal trajectory is defined in the sense that the control effort required to propel the drone should be minimized so as to ensure the most efficient usage of the onboard propulsion system.

A solution to this problem finds applications in numerous domains, package delivery being just one of them. Similarly, we can consider the idea of multi-agent surveillance and mapping of construction sites to find defects, where multiple aerial vehicles are deployed to scan and map a given establishment and they must avoid colliding with each other or into any other static obstacle. The wide range of applications of this problem served as a motivation to survey this topic.

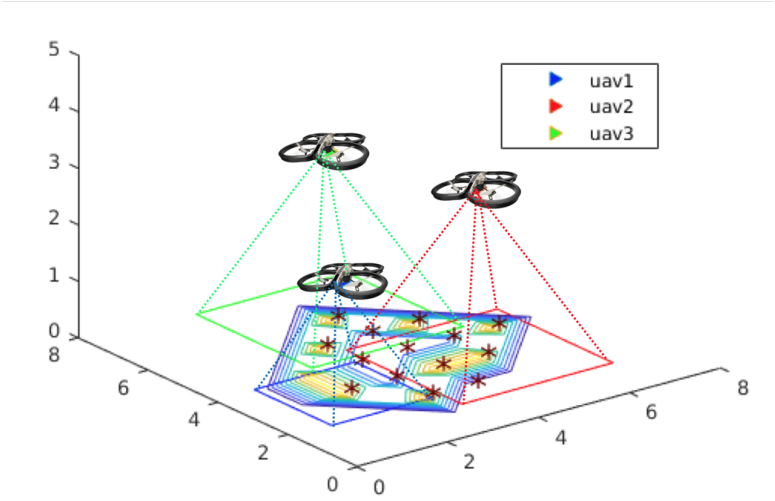


Figure 2: Using multiple UAVs for field coverage

1.2 Problem Description

We now formally describe the problem. Consider N drones with specified initial and final positions *i.e.*.

$$\mathbf{x}_i = \left[(x_{i1}, y_{i1}, z_{i1}), (x_{i2}, y_{i2}, z_{i2}), \cdots, (x_{iN}, y_{iN}, z_{iN})\right]$$

and,

$$\mathbf{x}_f = \left[(x_{f1}, y_{f1}, z_{f1}), (x_{f2}, y_{f2}, z_{f2}), \cdots, (x_{fN}, y_{fN}, z_{fN})\right]$$

we want to generate optimal trajectories for each drone so as to ensure that all the drones reach their final positions in a specified time T and these trajectories should not come closer than a specified radius of avoidance R, hence guaranteeing a safe collision-free trajectory generation.

1.3 Solution Architecture

An optimization-based approach is investigated to solve the problem described in the previous section. The trajectory dynamics are constructed using the acceleration of the UAV as the control input. We want to ensure that the generated trajectories require minimum control effort, therefore an optimization problem is formulated in order to minimize the norm squared of the acceleration of all the drones over the entire duration of the trajectory. The position, velocity, acceleration, and jerk constraints are formulated with convex inequalities, while the non-convex collision constraint inequality is approximated to a convex inequality and the resulting convex optimization problem is solved sequentially till convergence. Before formulating the problem mathematically, we will first review some basic ideas revolving around Sequential Convex Programming.

2 Sequential Convex Programming

Sequential Convex Programming (SCP) is a local optimization method for non-convex problems that leverage convex optimization. It is important to note that SCP happens to be a heuristic method and there do not exist theoretical guarantees that it can find an optimal (or even feasible) solution point. The results of the optimization algorithm using SCP can (and often do) depend on the initial point. However almost certainly or with some modifications, SCP often works well and finds a feasible point with a good enough, if not optimal, objective value.

2.1 Optimization Problem

We consider the non-convex problem

$$\min f_0(x)$$
$$s.t.\ f_i(x) \leq 0,\ \forall\, i = 1, 2, \ldots, m$$
$$h_j(x) = 0,\ \forall\, j = 1, 2, \ldots, p$$

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

2.2 Basic Idea of SCP

In the k^{th} iteration of SCP, maintain an estimate $x^{(k)}$ of the solution, and a convex trust region $\tau^{(k)} \subset \mathbb{R}^n$. Construct convex approximations $\bar{f}_0$, $\bar{f}_i$ of f_0 and f_i respectively and affine approximations $\bar{h_j}$ of h_j over the trust region $\tau^{(k)}$. The approximate convex problem so obtained is written as :

$$\min \bar{f}_0(x)$$
$$s.t.\ \bar{f}_i(x) \leq 0,\ \forall\, i = 1, 2, \ldots, m$$
$$\bar{h_j}(x) = 0,\ \forall\, j = 1, 2, \ldots, p$$

The solution to this approximate problem is $x^{(k+1)}$. Before we inspect some of these approximation techniques, we first understand the significance of the trust region.

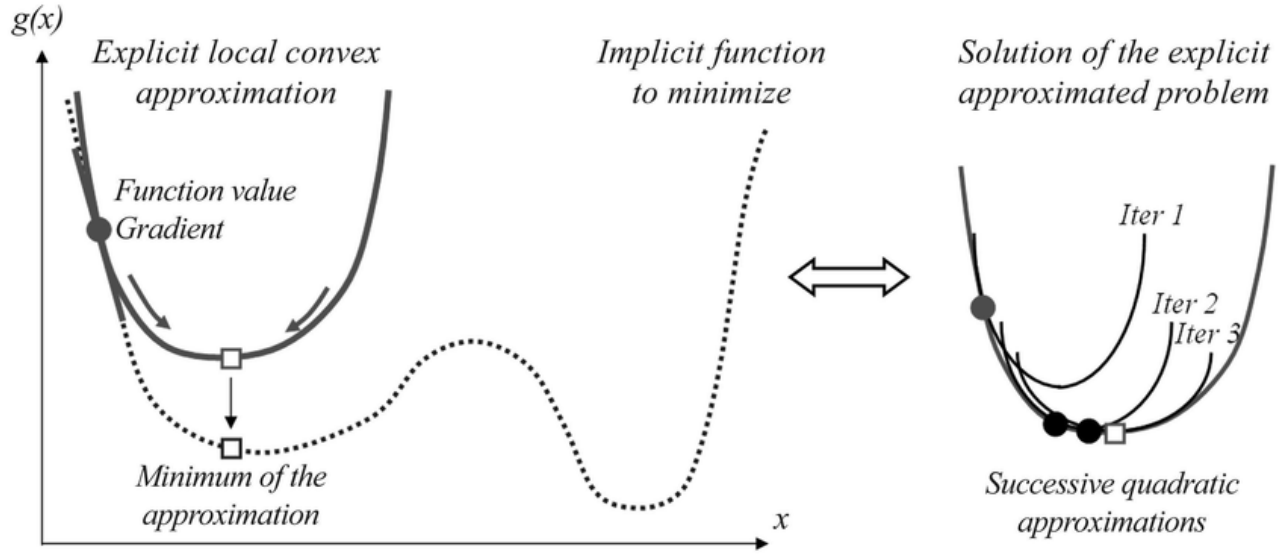


Figure 3: The fundamental idea of Sequential Convex Programming

2.3 Trust Region

The trust region at the k^{th} iteration is a neighborhood of the current solution estimate $x^{(k)}$, typically we define the trust region to be a box around the current solution estimate, *i.e.*

$$\tau^{(k)} = \{x; |x - x^{(k)}| < \rho\}$$

here ρ denotes the trust region radius. The trust region is manipulated based on the current estimate and the solution of the optimization problem at the current iteration. With this understanding of the trust region, we proceed to study some approximation techniques that can be utilized for the implementation of SCP.

2.4 Approximation Methods

In this section, we study three approximation techniques, *Taylor Approximation*, *Particle Method*, and *Quasi-linearization* as a piece of machinery for SCP.

2.4.1 Taylor Approximation

The affine first-order approximation of a function $f(x)$ using Taylor expansion is given by,

$$\bar{f}(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T (x - x^{(k)})$$

meanwhile, the second-order Taylor approximation is given by,

$$\bar{f}(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T (x - x^{(k)}) \frac{1}{2} (x - x^{(k)})^T P (x - x^{(k)})$$

Where $P = (\nabla^2 f(x^{(k)}))_+$ denotes the positive semi-definite part of the Hessian matrix. Note that in order for the resulting approximation to be convex, we must only consider the positive semi-definite component of the quadratic coefficient matrix. Note that the Taylor approximation provides us a **local** model which does not depend on the trust region radius ρ.

2.4.2 Particle Method

We start by randomly sampling points $z_1, z_2, \ldots, z_N \in \tau^{(k)}$, we then proceed to evaluate the function values at these points, *i.e.* $y_i = f(z_i)$. We then fit these data points (z_i, y_i) with a convex (or affine) function using convex optimization.

- For an affine approximation $\bar{h}(x)$ of a function $h(x)$, we first parameterize our hypothesis function by $\bar{h}(x) = \theta_0 + \theta_1 x$ and then formulate a least squares optimization problem:

$$\min_{\theta_0,\theta_1} \sum_{i=1}^{N} ||\bar{h}(z_i - x^{(k)}) - y_i||^2$$

  where the optimization problem is solved over the parameters θ_0, θ_1 of $\bar{h}(x)$.

- For a quadratic approximation to this data, we formulate an optimization problem,

$$\begin{aligned} \min_{P,q,r} \quad & \sum_{i=1}^{N} \left(\left(z_i - x^{(k)}\right)^T P \left(z_i - x^{(k)}\right) + q^T \left(z_i - x^{(k)}\right) + r \right)^2 \\ \text{s.t.} \quad & P \succeq 0 \text{ (positive semi-definiteness), where } P \in \mathbb{R}^{n\times n}, q \in \mathbb{R}^n, \text{ and } r \in \mathbb{R}. \end{aligned}$$

  This approach to approximation has an edge over the traditional Taylor approximation because *(i)* it can handle non-differentiable functions (or functions for which computing the derivatives is difficult) quite easily, *(ii)* this approach provides us with a **regional** model which depends on the current estimate $x^{(k)}$ as well as the trust-region radius ρ.

2.4.3 Qausi-linearization

A simple and cheap method for affine approximation. Consider the function $h(x)$,

$$h(x) = \frac{1}{2}x^T P x + q^T x + r$$

$$\implies h(x) = \left(\frac{1}{2}Px + q\right)^T x + r$$

as opposed to Taylor approximation, where the function is approximated around a point $x^{(k)}$, quasi-linearization allows for the coefficient matrix to be dependent only on the point $x^{(k)}$. To better illustrate, denote the affine Taylor approximation of $h(x)$ as $\bar{h}_T$ and the affine quasi-linear approximation as $\bar{h}_Q$, then the Taylor affine approximation is given by,

$$\bar{h}_T = (Px^{(k)} + q)^T (x - x^{(k)}) + r$$

whereas the quasi-linear approximation will be given by,

$$\bar{h}_Q = \left(\frac{1}{2}Px^{(k)} + q\right)^T x + r$$

It is evident Quasi-linearization and Taylor's approximation are analytical approximations and hence are computationally faster which comes at the cost of being a local model instead of a regional model. Whereas, we obtain a regional model in the Particle Method but it is computationally slower than Taylor or Quasi-linearization approaches, because of its sampling and optimization-based nature at every iteration. For most practical applications, Taylor approximation suffices, since it provides us with a computationally faster approach along with reasonably good accuracy.

2.5 Trust Region Update

In SCP, a trust region is a region around the current estimate $x^{(k)}$ where the objective function is well approximated by a quadratic function. The trust region constraint ensures that the iterates remain close to the current estimate so that the quadratic approximation remains accurate.

Updating the trust region is important in SCP because it determines the size of the region in which the quadratic approximation of the objective function is valid. If the trust region is too small, the algorithm may converge slowly or even stagnate at a local minimum. On the other hand, if the trust region is too large, the quadratic approximation may not be accurate, and the algorithm may diverge or produce poor-quality solutions.

Therefore, updating the trust region in SCP is a balancing act between exploration and exploitation. It is important to ensure that the trust region is large enough to allow for sufficient exploration of the search space, but not so large that the algorithm diverges or produces poor-quality solutions. Properly updating the trust region can lead to faster convergence and better solutions. In this section, we present an algorithm to update the trust region.

At the k^{th}, the trust region update algorithm judges the optimization progress by computing the decrease in the objective function f_0 using the solution $\tilde{x}$ of the explicit approximate optimization problem. We define the predicted decrease as $\bar{\delta} = f_0(x^{(k)}) - \bar{f}_0(\tilde{x})$ and the actual decrease $\delta = f_0(x^{(k)}) - f_0(\tilde{x})$.

With the notations well defined, we look into the trust region update algorithm.

Algorithm 1 Trust Region Update

At iteration k,

Require: $\alpha, \beta^f \in (0,1)$, $\beta^s \geq 1$ and $\tilde{x}$

$\bar{\delta} = f_0(x^{(k)}) - \bar{f}_0(\tilde{x})$
$\delta = f_0(x^{(k)}) - f_0(\tilde{x})$
if $\delta \geq \alpha\bar{\delta}$ **then**
    $\rho^{(k+1)} = \beta^s \rho^{(k)}$
    $x^{(k+1)} = \tilde{x}$
else $\{\delta < \alpha\bar{\delta}\}$
    $\rho^{(k+1)} = \beta^f \rho^{(k)}$
    $x^{(k+1)} = x^{(k)}$
end if

Crumbled into words, if the actual decrease is greater than α times the predicted decrease $\bar{\delta}$, the trust region size is increased, and the solution $\tilde{x}$ is accepted as a valid solution $x^{(k+1)}$. However, if the actual decrease is lesser than α times the predicted decrease, then the trust region size is decreased and the solution for the previous iteration $x^{(k)}$ is accepted as the solution of the current iteration.

The typical values of these constants are $\alpha = 0.1$, $\beta^s = 1.1$, and $\beta^f = 0.5$.

3 Mathematical Formulation

Now that we have understood the underlying machinery to solve this problem, we begin with the mathematical formulation of the multiagent trajectory optimization problem.

Given a set of initial and final positions for N drones, the objective is to ensure that these drones reach their respective desired positions within a specified time interval T while maintaining a minimum distance of at least R from each other throughout their trajectory. To achieve this objective, an optimization problem is formulated, which includes various convex and affine constraints, along with a non-convex collision avoidance constraint. The solution to this optimization problem guarantees the fulfillment of the stated objective.

3.1 Discretization

In order to solve this problem computationally, it is important to discretize the problem with an appropriate discretization factor h (chosen to be a small number). With this discretization factor, we obtain $K = \lceil \frac{T}{h} \rceil$ discrete temporal points. The value of discretization factor h is an important hyperparameter and significantly affects the solution trajectory, a small value of h will result in a finer trajectory, whereas a larger value of h will give a coarse trajectory.

3.2 Trajectory Dynamics

Let $p_i[k], v_i[k]$, and $a_i[k]$ denote the position, velocity and acceleration respectively of the i^{th} vehicle at discrete times $k \in \{1, 2, \ldots, K\}$, the trajectory is therefore governed by the following discretized dynamics,

$$v_i[k+1] = v_i[k] + ha_i[k] \tag{1}$$

$$p_i[k+1] = p_i[k] + hv_i[k] + \frac{h^2}{2}a_i[k] \tag{2}$$

which can be iteratively expanded to be written explicitly as a function of the accelerations of the vehicle at the previous time steps, therefore the position and velocity of vehicle i at time step $k(> 1)$ will be given by,

$$v_i[k] = v_i[1] + h(a_i[1] + a_i[2] + \cdots + a_i[k-1]) \tag{3}$$

$$p_i[k] = p_i[1] + h(k-1)v_i[1] + \frac{h^2}{2}((2k-3)a_i[1] + \cdots + a_i[k-1]) \tag{4}$$

for our implementation, we assume that the drones start from the initial position at rest, which implies that for all $i \in \{1, 2, \ldots, N\}$, $v_i[1] = 0$. This simplifies our iterative expansion as,

$$v_i[k] = h(a_i[1] + a_i[2] + \cdots + a_i[k-1]) \tag{5}$$

$$p_i[k] = p_i[1] + \frac{h^2}{2}((2k-3)a_i[1] + \cdots + a_i[k-1]) \tag{6}$$

from (5) and (6), we can express the position and velocity of all the drones at all times using only the accelerations in the previous time steps. Therefore we choose our **optimization variable** $\chi \in \mathbb{R}^{3NK}$ to consist of the accelerations of all the UAVs at each time step, *i.e.*,

$$\chi = \big[a_1[1], \cdots, a_1[K], \cdots, a_N[1], \cdots, a_N[K]\big]^T$$

3.3 Objective Function

For the trajectory dynamics (5) and (6) we chose the acceleration as the input. This choice intuitively makes sense, since the acceleration is proportional to the thrust force generated. And since we want to minimize the control efforts, we choose to optimize over the norm-squared of the acceleration of all the vehicles at all time stamps.

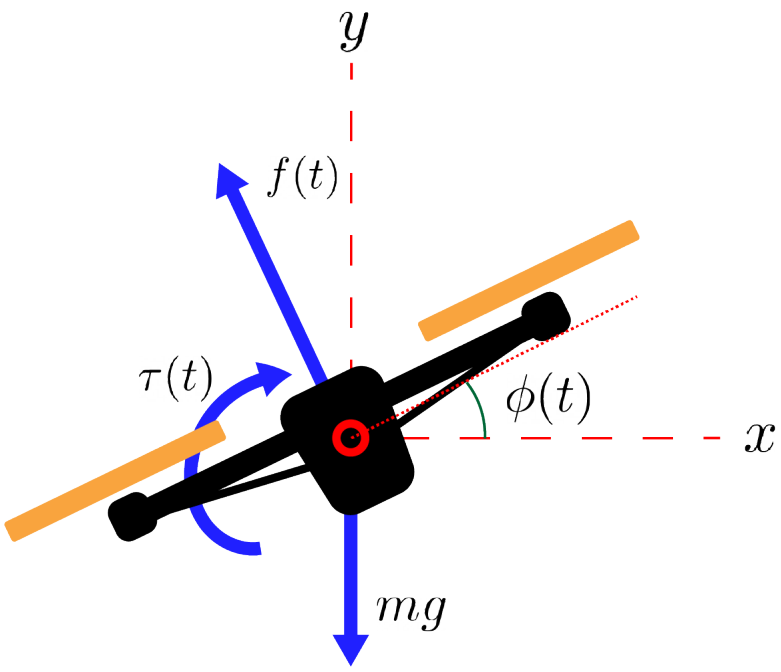


Figure 4: An accelerating 2D quadcopter

We, therefore, define our objective function to be,

$$f_0 = \sum_{i=1}^{N}\sum_{k=1}^{K} ||a_i[k] + \vec{g}||_2^2 \tag{7}$$

where $\vec{g} = \begin{bmatrix}0, 0, -9.81\end{bmatrix}$ is the gravity vector. For a single vehicle i at time k, define β_{ik} as,

$$\beta_{ik} = ||a_i[k] + \vec{g}||_2^2 = (a_i[k] + \vec{g})^T(a_i[k] + \vec{g})$$
$$\implies \beta_{ik} = a_i[k]^T a_i[k] + 2\vec{g}^T a_i[k] + \vec{g}^T\vec{g}$$

therefore the objective function may be re-written as,

$$f_0 = \sum_{i=1}^{N}\sum_{k=1}^{K} \beta_{ik}$$

$$\implies f_0 = \sum_{i=1}^{N}\sum_{k=1}^{K} \left(a_i[k]^T a_i[k] + 2\vec{g}^T a_i[k] + \vec{g}^T\vec{g}\right)$$

since $\vec{g}^T\vec{g} = (-9.81)^2$ which is a constant, we drop off this term and re-write the objective function as,

$$\implies f_0 = \sum_{i=1}^{N}\sum_{k=1}^{K} \left(a_i[k]^T a_i[k] + 2\vec{g}^T a_i[k]\right)$$

upon expanding out the summation we get,

$$f_0 = \big(a_1[1]^T a_1[1] + 2\vec{g}^T a_1[1]\big) + \big(a_1[2]^T a_1[2] + 2\vec{g}^T a_1[2]\big) + \cdots + \big(a_1[K]^T a_1[K] + 2\vec{g}^T a_1[K]\big) + \cdots$$
$$+ \big(a_N[1]^T a_N[1] + 2\vec{g}^T a_N[1]\big) + \big(a_N[2]^T a_N[2] + 2\vec{g}^T a_N[2]\big) + \cdots + \big(a_N[K]^T a_N[K] + 2\vec{g}^T a_N[K]\big)$$

re-arranging the terms,

$$f_0 = \big(a_1[1]^T a_1[1] + \cdots + a_1[K]^T a_1[K] + \cdots + a_N[1]^T a_N[1] + \cdots + a_N[K]^T a_N[K]\big)$$
$$+ 2\vec{g}^T \left(a_1[1] + \cdots + a_1[K] + \cdots + a_N[1] + \cdots + a_N[K]\right)$$

we can represent this objective explicitly as a function of the optimization variable χ as follows,

$$f_0 = \chi^T P \chi + q^T \chi \tag{8}$$

with $P = I_{3NK}$ and $q = \left[2\vec{g}, 2\vec{g}, \cdots, 2\vec{g}\right]^T \in \mathbb{R}^{3NK}$, where I_{3NK} denotes the identity matrix of size $3NK \times 3NK$. Since I_{3NK} is a positive definite matrix, we conclude that the functional form of the objective function obtained in (8) is a strictly convex quadratic form in χ.

3.4 Constraints

In order to achieve the desired objective, there are various constraints that must be enforced along with the minimization of the accelerations. The initial and final positions of all the vehicles are given, Furthermore, to meet a vehicle's physical constraints, velocity, acceleration, and jerk can be constrained. Lastly, we need to ensure that the generated trajectories are at least a distance R apart from each other at all time steps, hence guaranteeing collision avoidance.

We shall now formulate these constraints and represent them in a matrix form, this representation will be easier if we consider only one vehicle at a time and later we will combine these constraints to reconstruct the constraints for all the vehicles. Let $\chi_i \in \mathbb{R}^{3K}$ denote the accelerations at all times for the vehicle i $\Big(\implies \chi_i = \left[a_i[1], a_i[2], \cdots, a_i[K]\right]^T\Big)$.

3.4.1 Equality Constraints

To commence, we shall specify the affine equality constraints. It is noted that the UAVs are initially at rest, thus their initial velocities and accelerations are zero. As they approach their respective goals, they are required to decelerate until they come to a complete stop. As such, their final velocities and accelerations must also be zero. In addition to these velocity and acceleration constraints, we possess knowledge of the initial and final positions of all UAVs, and the optimization algorithm must accordingly uphold these conditions by imposing appropriate constraints. However, observe that imposing the trajectory dynamics (5) and (6) automatically imposes the initial position and initial velocity constraint, therefore we only need to explicitly impose the equality constraints on the initial and final accelerations as well as the final positions and velocities.

- Final Position Constraint: Since the final positions $\mathbf{x}_f$ are given, for a single vehicle i we have that,

$$p_i[K] = \mathbf{x}_f[i]$$
$$\implies p_i[1] + \frac{h^2}{2}((2K-3)a_i[1] + \cdots + a_i[K-1]) = \mathbf{x}_f[i]$$

therefore, the final position equality constraint for vehicle i is given by,

$$(2K-3)a_i[1] + \cdots + a_i[K-1] = \frac{2}{h^2}(\mathbf{x}_f[i] - p_i[1]) \tag{9}$$

the matrix representation of this constraint will be written as,

$$A_{eq,fp,i}\chi_i = b_{eq,fp,i} \tag{10}$$

where $A_{eq,fp,i} \in \mathbb{R}^{3\times 3K}, b_{eq,fp,i} \in \mathbb{R}^3$.

- Final Velocity Constraint: We want the UAVs to come to rest as they reach their goal points, therefore for a single vehicle i we have that,

$$v_i[K] = \mathbf{0}_{3\times 1}$$

this implies that the final velocity constraint for vehicle i is given by,

$$a_i[1] + a_i[2] + \cdots a_i[K-1] = \mathbf{0}_{3\times 1} \tag{11}$$

the matrix representation of this constraint will be written as,

$$A_{eq,fv,i}\chi_i = \mathbf{0}_{3\times 1} \tag{12}$$

where $A_{eq,fv,i} \in \mathbb{R}^{3\times 3K}$.

- Final Acceleration Constraint: Since the final velocities are 0, we trivially conclude that the final accelerations are also 0. Therefore, for a single vehicle i we have that,

$$a_i[K] = \mathbf{0}_{3\times 1} \tag{13}$$

the matrix representation of which will be written as,

$$A_{eq,fa,i}\chi_i = \mathbf{0}_{3\times 1} \tag{14}$$

where $A_{eq,fa,i} \in \mathbb{R}^{3\times 3K}$.

- Initial Acceleration Constraint: Since the initial velocities are imposed to be zero from the trajectory dynamics (5) and (6), we conclude that the initial acceleration for all vehicles is trivially 0. Therefore for a single vehicle i we have that,

$$a_i[1] = \mathbf{0}_{3\times 1} \tag{15}$$

representing this constraint in matrix terms,

$$A_{eq,ia,i}\chi_i = \mathbf{0}_{3\times 1} \tag{16}$$

where $A_{eq,ia,i} \in \mathbb{R}^{3\times 3K}$.

Now that we have defined the equality constraint matrices for a single vehicle, we need to assemble these equality constraints into a single matrix. For a single vehicle i, all these equality constraints can be stacked on top of each other and therefore the equality constraint matrix representation is written as,

$$A_{eq,i}\chi_i = b_{eq,i} \tag{17}$$

with,

$$A_{eq,i} = \begin{bmatrix} A_{eq,fp,i} \\ A_{eq,fv,i} \\ A_{eq,fa,i} \\ A_{eq,ia,i} \end{bmatrix} \in \mathbb{R}^{12\times 3K}$$

and,

$$b_{eq,i} = \begin{bmatrix} b_{eq,fp,i} \\ \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \end{bmatrix} \in \mathbb{R}^{12}$$

We have now constructed the equality constraint matrices for a single vehicle, we extend this idea to represent the total equality constraint matrices for all the vehicles. To obtain the total equality constraint matrices, we vertically stack the accelerations for a single vehicle χ_i to obtain χ, to ensure that the matrix multiplication operations are consistent, we need to stack the coefficient matrices diagonally and the residual matrices vertically.

$$A_{eq} = \begin{bmatrix} A_{eq,1} & & \\ & \ddots & \\ & & A_{eq,N} \end{bmatrix},\ b_{eq} = \begin{bmatrix} b_{eq,1} \\ \vdots \\ b_{eq,N} \end{bmatrix}$$

$$\implies A_{eq}\chi = b_{eq} \tag{18}$$

All other terms apart from the diagonal entries in A_{eq} are 0. This concludes the construction of the equality constraint matrices. Observe that all the equality constraints are affine.

3.4.2 Inequality Constraints

The UAVs will be operating in a real physical space, where there are many safety and dynamic constraints that need to be enforced via inequalities. We analyze these constraints in this section.

- Position Constraints: The workspace is physically restricted by some p_{max} and p_{min}, therefore for a single vehicle i at time step $k(>1)$, we have that,

  $$p_{min} \leq p_i[k] \leq p_{max}$$

  from the trajectory dynamics, we have that,

  $$\frac{2}{h^2}(p_{min} - p_i[1]) \leq (2k-3)a_i[1] + (2k-5)a_i[2] + \ldots + a_i[k-1] \leq \frac{2}{h^2}(p_{max} - p_i[1]) \tag{19}$$

  this inequality must hold true for all values of $k \in \{2, 3, \ldots, K\}$. Therefore the resulting matrix representation will be,

  $$A_{in,p,i}\chi_i \leq b_{in,p,i} \tag{20}$$

  where $A_{in,p,i} \in \mathbb{R}^{6(K-1)\times 3K}$, $b_{in,p,i} \in \mathbb{R}^{6(K-1)}$.

- Velocity Constraints: The velocity of the UAVs must be restricted by v_{min} and v_{max} so as to ensure safe flight in a confined space as well as dynamic feasibility, therefore for a single vehicle i at time step $k(>1)$, we have that,

$$v_{min} \leq v_i[k] \leq v_{max}$$

using the trajectory dynamics, we express this inequality in terms of the accelerations,

$$\frac{v_{min}}{h} \leq a_i[1] + a_i[2] + \ldots + a_i[k-1] \leq \frac{v_{max}}{h} \tag{21}$$

this inequality must hold true for all values of $k \in \{2, 3, \ldots, K\}$, and thus the matrix representation can be written as,

$$A_{in,v,i}\chi_i \leq b_{in,v,i} \tag{22}$$

where $A_{in,v,i} \in \mathbb{R}^{6(K-1)\times 3K}$, $b_{in,v,i} \in \mathbb{R}^{6(K-1)}$.

- Acceleration Constraints: For the trajectory to be smooth and dynamically feasible, we need to constrain the acceleration of all the vehicles at all times between some a_{min} and a_{max}, therefore for a single vehicle i at time step $k(>1)$, we have that,

$$a_{min} \leq a_i[k] \leq a_{max} \tag{23}$$

this inequality must hold true for all values of $k \in \{2, 3, \ldots, K\}$ and thus the matrix representation is written as,

$$A_{in,a,i}\chi_i \leq b_{in,a,i} \tag{24}$$

where $A_{in,a,i} \in \mathbb{R}^{6(K-1)\times 3K}$, $b_{in,a,i} \in \mathbb{R}^{6(K-1)}$.

- Jerk Constraints: Jerk is defined as the third derivative of position, owing to the discretization of the problem we can define jerk as the derivative of acceleration, so at time $k(>1)$ for a single vehicle i, we have that,

$$j_i[k] = \frac{a_i[k] - a_i[k-1]}{h}$$

for a dynamically feasible and smooth trajectory, we need to ensure that the jerk experienced by the UAV at all times is bounded by j_{min} and j_{max}, therefore the resulting inequality is,

$$j_{min} \leq j_i[k] \leq j_{max}$$

which in turn implies,

$$hj_{min} \leq a_i[k] - a_i[k-1] \leq hj_{max} \tag{25}$$

the matrix representation will be written as,

$$A_{in,j,i}\chi_i \leq b_{in,j,i} \tag{26}$$

where $A_{in,j,i} \in \mathbb{R}^{6(K-1)\times 3K}$, $b_{in,j,i} \in \mathbb{R}^{6(K-1)}$.

The inequality constraints developed so far form a convex feasible set, however, the collision avoidance constraint is defined as a non-convex constraint. We use the literature of SCP to obtain an affine approximation of this constraint and then we shall construct the total inequality matrices for the true optimization variable χ.

- Collision Avoidance Constraint: For safe trajectory generation, the distance between any two vehicles $i, j (i \neq j)$ at all times $k \in \{2, 3, \ldots, K\}$ should be at least R,

$$||p_i[k] - p_j[k]||_2^2 \geq R$$

at iteration $(q+1)$ of the optimization algorithm, using the first-order Taylor approximation, we obtain an affine approximation of the collision avoidance constraint around the previous solution χ^q,

$$||p_i^q[k] - p_j^q[k]||_2 + \eta^T \left[(p_i[k] - p_j[k]) - (p_i^q[k] - p_j^q[k])\right] \geq R \tag{27}$$

where,

$$\eta = \frac{p_i^q[k] - p_j^q[k]}{||p_i^q[k] - p_j^q[k]||_2}$$

define $\alpha_{ij}^q[k] = p_i^q[k] - p_j^q[k]$, the constraint (27) can be written as,

$$||\alpha_{ij}^q[k]||_2 + \eta^T \left[(p_i[k] - p_j[k]) - \alpha_{ij}^q[k]\right] \geq R$$

but,

$$\eta = \frac{\alpha_{ij}^q[k]}{||\alpha_{ij}^q[k]||_2}$$

$$\implies ||\alpha_{ij}^q[k]||_2 + \frac{\alpha_{ij}^q[k]^T}{||\alpha_{ij}^q[k]||_2} \left[(p_i[k] - p_j[k]) - \alpha_{ij}^q[k]\right] \geq R$$

which can be simplified into,

$$\alpha_{ij}^q[k]^T (p_i[k] - p_j[k]) \geq R||\alpha_{ij}^q[k]||_2 \tag{28}$$

using the trajectory dynamics we substitute the values of $p_i[k]$ and $p_j[k]$ to get,

$$\alpha_{ij}^q[k]^T \left(\left(p_i[1] + \frac{h^2}{2} \left((2k-3)a_i[1] + \cdots + a_i[k-1]\right) \right) - \left(p_j[1] + \frac{h^2}{2} \left((2k-3)a_j[1] + \cdots + a_j[k-1]\right) \right) \right) \geq R||\alpha_{ij}^q[k]||_2$$

which implies,

$$\begin{aligned} &\alpha_{ij}^q[k]^T \left(((2k-3)a_i[1] + \cdots + a_i[k-1]) - ((2k-3)a_j[1] + \cdots + a_j[k-1])\right) \\ &\geq \frac{2}{h^2} \left(R||\alpha_{ij}^q[k]||_2 - \alpha_{ij}^q[k]^T (p_i[k] - p_j[k])\right) \end{aligned} \tag{29}$$

This inequality must hold true for all values of $k \in \{2, 3, \ldots, K\}$, therefore the matrix representation of this inequality for two vehicles $i, j (i \neq j)$ will be written as,

$$\alpha_{ij}^q[k]^T A_{in,col,i,j} \chi \leq b_{in,col,i,j} \tag{30}$$

where $A_{in,col,i,j} \in \mathbb{R}^{3 \times 3NK}, b_{in,col,i,j} \in \mathbb{R}$. Observe that there will be total of ${}^N C_2 (K-1) = \frac{N(N-1)}{2}(K-1)$ collision constraints. Denote the total collision avoidance constraint matrices by $A_{in,col} \in \mathbb{R}^{{}^N C_2 (K-1) \times 3NK}, b_{in,col} \in \mathbb{R}^{{}^N C_2 (K-1)}$

Now that we have studied the inequality constraints for a single vehicle, we need to assemble them to form the total inequality constraint matrices.

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

$$A_{in,i} = \begin{bmatrix} A_{in,p,i} \\ A_{in,v,i} \\ A_{in,a,i} \\ A_{in,j,i} \end{bmatrix} \in \mathbb{R}^{24(K-1)\times 3K}$$

and,

$$b_{in,i} = \begin{bmatrix} A_{in,p,i} \\ b_{in,v,i} \\ b_{in,a,i} \\ b_{in,j,i} \end{bmatrix} \in \mathbb{R}^{24(K-1)}$$

Therefore the total inequality matrices for the first iteration are constructed by stacking the residual matrices for all vehicles vertically, whereas to ensure consistency according to matrix multiplication, the coefficient matrices for all the vehicles are stacked diagonally.

$$A_{1,in} = \begin{bmatrix} A_{in,1} & & \\ & \ddots & \\ & & A_{in,N} \end{bmatrix},\ b_{1,in} = \begin{bmatrix} b_{in,1} \\ \vdots \\ b_{in,N} \end{bmatrix}$$

$$\implies A_{1,in}\chi \leq b_{1,in} \tag{31}$$

The solution of the first iteration of the optimization loop is denoted by χ^0 and from the next iteration onwards, the collision constraint is enforced with linearization around the previous solution.

- In the first iteration, we do not have any previous solution to linearize our collision constraint, therefore we do not consider the collision constraint in the first iteration. However, from the **second iteration onwards** we can enforce the collision avoidance constraints since we now have a previous solution around which we can linearize the collision avoidance constraint. The total inequality constraint matrices are constructed by vertically appending $A_{in,col}$ and $b_{in,col}$ to $A_{1,in}$ and $b_{1,in}$ obtained in (31) respectively. Therefore we get,

$$A_{in} = \begin{bmatrix} A_{1,in} \\ A_{in,col} \end{bmatrix},\ b = \begin{bmatrix} b_{1,in} \\ b_{in,col} \end{bmatrix}$$

$$A_{in}\chi \leq b_{in} \tag{32}$$

3.5 Implementation

With the derived objective function (8), equality constraints (18), and the approximated inequality constraints (32). The resulting problem has a convex quadratic objective function over a convex set,

hence forming a Quadratic Programming problem.

$$\begin{aligned} \min \quad & \chi^T P \chi + q^T \chi \\ \text{subject to} \quad & A_{eq}\chi = b_{eq} \\ & A_{in}\chi \leq b_{in} \end{aligned} \tag{33}$$

QPs can be efficiently solved using readily available libraries, one such example is Python's CVXOPT library. With an understanding of each individual component of the optimization problem, we proceed to compactly lay down the exact optimization algorithm.

Algorithm 2 Multiagent Trajectory Optimization using SCP

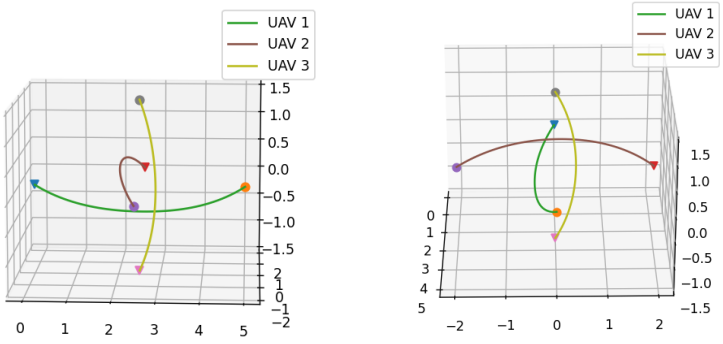
Require: $\mathbf{x}_i, \mathbf{x}_f, N, R, T, m_i, \epsilon, h, p_{max}, p_{min}, v_{max}, v_{min}, a_{max}, a_{min}, j_{max}, j_{min}$

$\chi^0 \leftarrow QP(P, q, A_{eq}, b_{eq}, A_{1,in}, b_{1,in})$
$f_p \leftarrow f(\chi^0)$
$n_i \leftarrow 0$
while $n_i \leq m_i$ **do**
  $\chi \leftarrow QP(P, q, A_{eq}, b_{eq}, A_{in}, b_{in})$
  $f_c \leftarrow f(\chi)$
  if $|f_c - f_p| < \epsilon$ **then**
    break
  end if
  $f_p \leftarrow f_c$
  $n_i \leftarrow n_i + 1$
end while

Algorithm 2 is implemented using Python, and the **link** for the code is provided.

4 Experiments

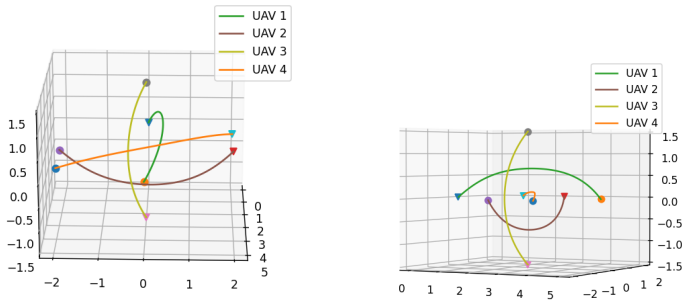
- **Experiment 1:**



(a) (b)

Figure 5: Mutual perpendicular motion for 3 UAVs

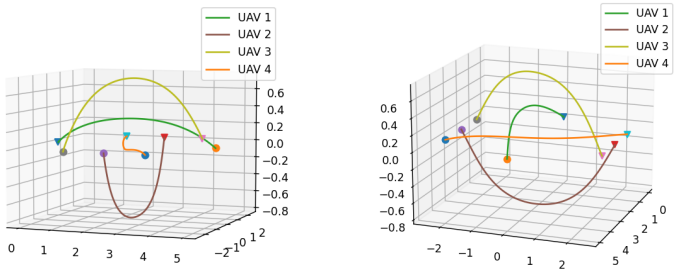
- **Experiment 2:**



(a) (b)

Figure 6: 3 UAVs moving perpendicularly, 1 moving diagonally

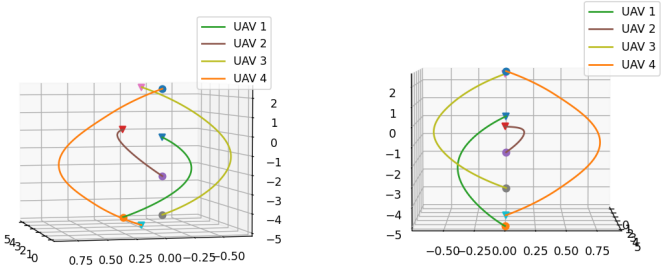
- **Experiment 3:**



(a) (b)

Figure 7: 3 UAVs moving towards one plane and 1 UAV crossing their paths

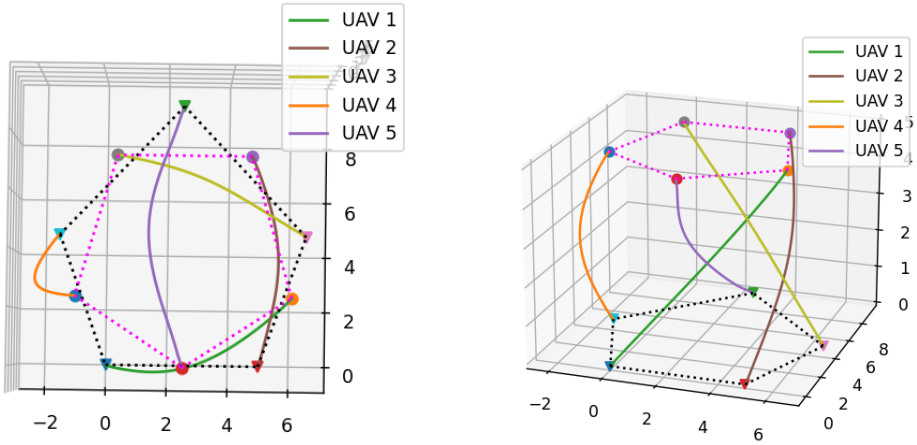
- **Experiment 4:**



(a) (b)

Figure 8: 4 UAVs having start and end points in the same plane with crossing baseline trajectories

- **Experiment 5:**

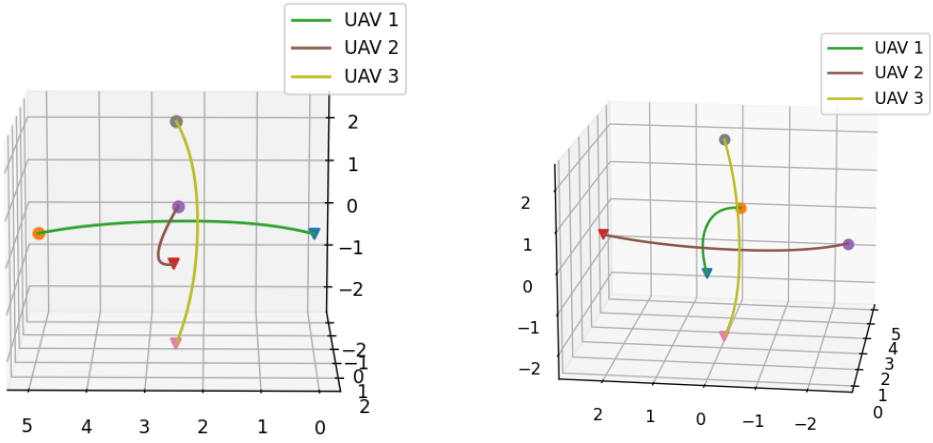


(a) (b)

Figure 9: Application to formation flight

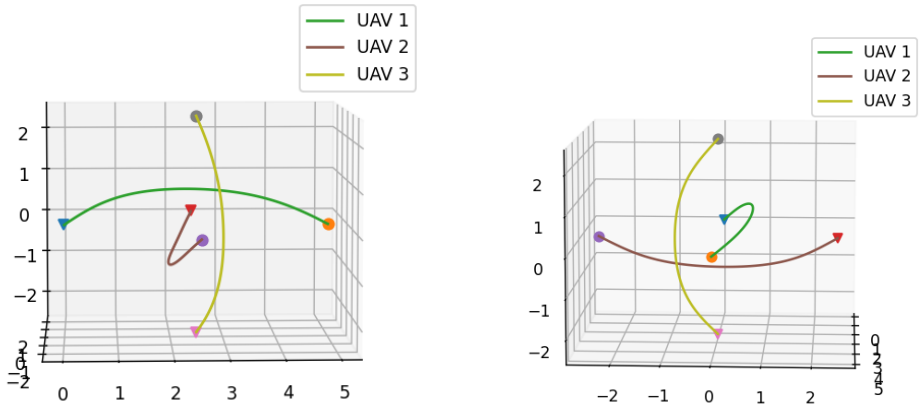
5 Analysis

In the previous section, we saw some experiments clearly the optimization algorithm generates smooth collision-free trajectories, from these demonstrations we can see that this algorithm finds applications in many tasks such as mapping and surveillance, formation flight, etc. In this section, we will study the effect of the radius of avoidance parameter R on the generated trajectories. Theoretically increasing this parameter should push the trajectories away from each other allowing for a greater distance at all time steps. Similarly decreasing R should bring the trajectories closer to each other.



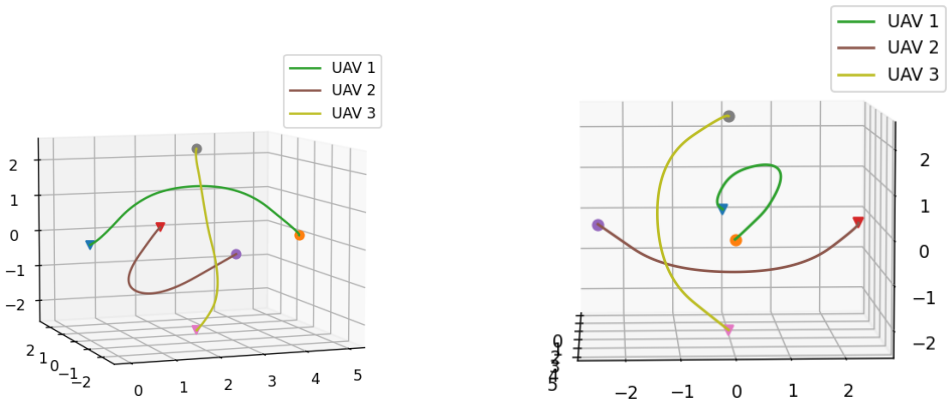
(a) (b)

Figure 10: $R = 0.5m$



(a) (b)

Figure 11: $R = 1.5m$



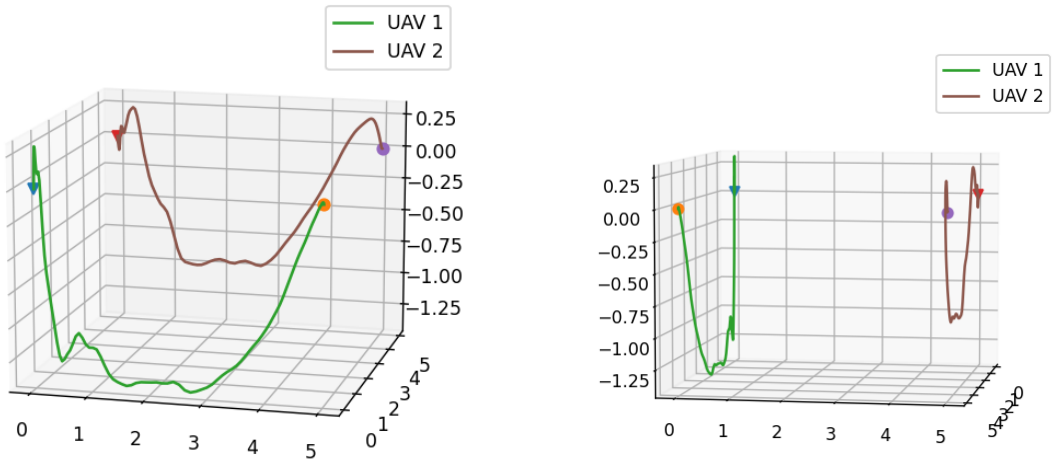
(a) (b)

Figure 12: $R = 2.5m$

As we can see from the trajectories generated above, increasing the value of R pushes the trajectories away from each other which serves as a benchmark to validate the algorithm and hence makes the experimental results consistent with our theoretical understanding.

6 Limitations

The current implementation of this algorithm deals with only those cases where the baseline trajectories (the straight line from the start point to the goal point) of at least 2 of the vehicles intersect which implies a possible collision. However, if the initial and final positions are selected such that the baseline trajectories do not intersect then in this case, a trajectory is indeed generated that respects the equality constraints but fails to generate an optimal-in-acceleration trajectory. This section presents some of the fail cases where peculiar trajectories are generated and possible causes are explored.



(a) (b)

Figure 13: Suboptimal trajectories generated since baseline trajectories do not intersect

Possible reasons for this failure include (i) the crawling problem (slight changes in the objective function in the initial stages of the optimization process leading to the satisfaction of the convergence criterion), (ii) SCP is only a heuristic and has no theoretical guarantees of global convergence, hence the result strongly depends on the initial solution χ^0. This problem can be easily avoided by hard-coding these specific cases cleverly, however, the real benchmark of this algorithm is its ability to deal with cases where it has to avoid obstacles which were successfully demonstrated in sections 4 and 5.

7 Conclusions

A multiagent trajectory planning problem was successfully formulated as an optimization problem with both convex and non-convex constraints. By leveraging the literature of Sequential Convex Programming, non-convex constraints were approximated and a locally convex problem was sequentially solved. The experimental results were thoroughly analyzed and aligned with theoretical expectations, validating the efficacy of Sequential Convex Programming in trajectory optimization problems. Furthermore, this approach can be easily extended to address similar problems and diverse types of vehicles, making it a versatile and practical solution for optimizing trajectories.

8 References

[1] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors”, in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2520-2525, IEEE, 2011.
[2] F. Augugliaro, A. P. Schoellig, and R. D’Andrea, “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ Internation Conference on.* IEEE, 2012, pp. 1917-1922.
[3] **Figure 1 source**
[4] **Figure 2 source**
[5] **Figure 3 source**
[6] **Figure 4 source**

9 Contributions

- **Mathematical Formulation:** Saad Khan
- **Code Implementation, Poster:** Saad Khan, Raj Kachhadiya, Ayush Avinash Tamgadge, Arun Kumar Meena (Poster Link)
- **Debugging, Report:** Saad Khan, Raj Kachhadiya
- **Testing:** Ayush Avinash Tamgadge, Arun Kumar Meena

10 Authors

- **Saad Khan** (200100085)
- **Raj Kachhadiya** (200100077)
- **Ayush Avinash Tamgadge** (20D100004)
- **Arun Kumar Meena** (200100034)