Optimal Path Smoothing while Maintaining a Region of Safe Operation

Justin Whitaker Brian Merrell

Abstract—
Multi-vehicle control scenarios often require controlling multiple vehicles to move together in coordinated formations. In addition to planning the overall motion of the formation, each vehicle must have a collision-free trajectory that the vehicle is capable of tracking. Work has been done that generates such trajectories, but these may have unnecessary and undesirable regions of high path curvature and/or vehicle acceleration. Typical path smoothing techniques, like splines, may result in trajectories that are no longer collision free. This work seeks to provide optimally smooth follower trajectories by the optimization of a trajectory according to safe region constraints.

I. Background

Formation control is considered a canonical problem in multi-vehicle controls. Methods of formation control include leader-follower, virtual structure and behavioral approaches. Virtual structures allow for paths to be generated for each vehicle based on the planned path of a leader vehicle. These paths, however, can include regions of high curvature and vehicle accelerations. These paths can be smoothed, but have no guarantee that the resulting follower vehicle paths are free of inter-vehicle collisions. Collision prevention can be done by defining regions of safe operation for each vehicle based on the structure of the formation. Voronoi tessellations allow for easily defining these safe regions of operation. This work seeks to smooth the follower paths according to the resulting time varying regions of safe operation. This smoothing of a single vehicle's path will be achieved by formulating the problem as an optimization of the path trajectory subject to the linear constraints of the safe region. The objective function will account for the desires to smooth the path, not stray too far from the nominal formation position, and to end close to the same nominal formation position. This allows the path to be optimally smooth with the guarantee that no vehicle will leave its region of safe operation, which guarantees the smoothed paths to be free of collisions.

While there are many approaches to formation control, three general approaches are particularly relevant to this work. First, in a virtual structure approach the entire formation is treated as a single entity. By defining a single oriented point within the structure, the desired position of each agent is defined [1]. Second, the leader-follower approach defines an agent as a leader that all other agents within the formation follow. This provides a centralized interface for planning a formation: an operator commands the leader and the formation follows. The first and second methods can be combined by defining a virtual leader (an artificial vehicle that an operator can control perfectly), which provides centralized structured movement. Third, a behavior-based approach achieves formation control by defining specific behaviors for individual agents. For instance, artificial potential fields can be designed and combined to have agents avoid each other, avoid obstacles, and/or converge to a desired position within the formation [2].

The virtual leader formation is of particular interest to this work. In this approach a structure is defined in terms of an offset vector for each vehicle from some reference point in the structure. By giving this reference point an orientation and dynamics, the movement of the structure (and the desired positions of each agent) can be controlled and planned. After this modification, the reference point is often called a virtual leader.

The static relative nature of virtual structures allows for regions of safe operation to be defined for each follower vehicle as well. The desired position of each follower can be used to form a Voronoi tessellation. The nature of Voronoi tessellations means that each resulting region only contains one vehicle's desired location. If each vehicle is then constrained to stay within its corresponding region, the vehicles are guaranteed to not collide with each other. This is a useful method of ensuring the vehicles remain collision free without requiring exact knowledge of the locations of the other vehicles [3].

Because of the definition of the structure in relation to the virtual leader, a follower vehicle's desired path can be generated based on a given path for the virtual leader. The offset vector from the leader position and orientation specifies a follower point for every point in the leader trajectory. It is straightforward, therefore, to generate induced paths for followers. Additionally, if the leader path is generated to have continuous curvature, or to be smooth, with a certain maximum curvature, then the follower paths will also have paths of continuous curvature, with bounded maximum curvature. While bounded, the maximum curvature of each follower is different in general from the leader path's maximum curvature. Additionally, these induced paths can require large accelerations (both translational and rotational) of the follower vehicle. For example, consider two follower vehicles, one on either side of the virtual leader. As the leader makes a turn, the outermost follower must undergo large accelerations to stay with the desired location. Turning the other direction, the other follower must also do the same.

This introduces the desire to have more "smooth" paths for the follower vehicles. Smoothness of path can take on a variety of definitions depending on the application. A "smooth"

trajectory often signifies that the trajectory belongs to the class $\mathcal{C}^k$ functions for some value of k, typically two or three (herein, trajectories are assumed to be continuous in time, and this continuity is not counted toward the class of a function). Dubins paths can be used to generate class $\mathcal{C}^1$ smooth curves of shortest path length between oriented waypoints. Being in $\mathcal{C}^1$, however, means that, while the velocity is continuous, there is a discontinuity in acceleration. In most cases, vehicles cannot actually execute a trajectory with this discontinuity in acceleration. Various methods have been used to generate class $\mathcal{C}^2$ trajectories (continuous in acceleration) including splines and clothoids [4].

Splines are curves that are defined piecewise with polynomials that are mutually smooth at the junctions [4]. Cubic splines are often used as a method of generating trajectories that are of class $\mathcal{C}^2$. Because cubic polynomials are themselves of class $\mathcal{C}^2$ (being ultimately of the class $\mathcal{C}^3$), any piecewise composition of cubic polynomials that have the same value of second derivative at the junction points is also of class $\mathcal{C}^2$. Cubic polynomials are the minimum degree polynomials for which class $\mathcal{C}^2$ functions can be formed piecewise, as at any lower degree the stipulation that the second derivative be equal at one point would imply that the two conjoining polynomials would be the same. Both interpolating (passing through waypoints) and smoothing (passing near waypoints) cubic splines can be one method of providing $\mathcal{C}^2$ smooth trajectories [4].

Spline methods can be extended to higher level classes of continuous functions by using a higher degree polynomial and require continuity at a deeper level of differentiation. Additionally, control theoretic splines can be used to achieve maximally smooth splines. Optimization techniques and ideas are used to form splines that trade off smoothness and proximity to waypoints in the optimal manner [5].

Another method of determining a class $\mathcal{C}^2$ is to build on top of a Dubins path and add clothoid transitions to ensure continuity of curvature. Clothoids are curves which have linearly varying curvature. Because a Dubins path transitions from a line, with no curvature, to an arc, with a constant curvature, back to a line, a clothoid is a straightforward method of transitioning between the arc and the lines. In addition, a maximum curvature constraint can be imposed on the trajectory so that it never exceeds that curvature constraint. The continuity in curvature ensures continuity in accelerations. It is this general method that [3] uses in generating a leader path, with the addition of zero-error trajectory tracking with epsilon-point control.

These methods of smoothing, however, do not always take into account the overall magnitude of the derivatives of a trajectory. A trajectory can be "smooth" insofar as it is a class $\mathcal{C}^2$ (or higher) function, but have high values of velocities which result in rapid changes in position. Even assuming that the vehicle is capable of actuating with these high levels of velocity, a path with high variation does not seem "smooth". Why should a vehicle go far to the left, then far to the right, only to then continue in a "middle" trajectory? Perhaps in some cases this is desirable or intended behavior, but most often it is not. This gives rise to another sense of a "smooth" trajectory: to limit the variation of the trajectory, or in other words, bound the derivative of the trajectory within some value.

For some applications, this concept of "smoothness" implying bounded velocity can be extended to include further derivatives. In some cases, there may be a desired velocity, acceleration, or lower derivative to be maintained by the vehicle. This may either be because of the vehicle (for example, a fixed wing aircraft has optimal operating speeds) or the task. The generalized case of this second sense of smoothness would then be to bound all applicable derivatives of the trajectory.

This work proposes to capture this second sense of smoothness, along with the first, by optimizing a follower trajectory constrained by integration dynamics (to ensure smoothness of the first sense) and the linear boundary constraints of the region of safe operation (to ensure collision free trajectories). This is achieved by minimizing the value of the derivatives of the trajectory position (providing smoothness of the second sense), in conjunction with the deviation from the nominal trajectory (that induced by the leader trajectory and the formation) according to the constraints listed above. The following section gives the details of the problem definition.

II. The Problem

The problem can be simplified to consider a single follower trajectory independent of any of the other follower trajectories. This can be done because the bounding region specifies a "safe" area that will never contain another vehicle, eliminating the need to consider the other vehicles when planning a trajectory if it is assumed that each vehicle never leaves its bounding region. Thus, all aspects of this formulation will be given without specifying a particular vehicle, but will be applicable to each vehicle. Additionally, it is assumed that the dynamics of the vehicles in question are differentially flat, indicating that, in some way, the generated trajectory (perhaps with some of its derivatives) can be used to determine the vehicle states.

The optimization problem can be stated as a discrete, multi-stage optimization of $\mathbf{y}$ (a composition of the state $\mathbf{x}$ and the input $\mathbf{u}$),

$$\mathbf{y} = \begin{bmatrix}\mathbf{x}_0 & \mathbf{u}_0 & \dots & \mathbf{x}_{N-1} & \mathbf{u}_{N-1} & \mathbf{x}_N\end{bmatrix}^T, \tag{1}$$

as follows

$$\begin{aligned} &\min_{\mathbf{y}} \ J(\mathbf{y}) \\ &s.t. \\ &\quad A_c\mathbf{y} \leq b \\ &\quad \mathbf{x}_{k+1} = \bar{A}\mathbf{x}_k + \bar{B}\mathbf{u}_k; \quad \mathbf{x}_0 = \mathbf{x}(t_0); \\ &\quad \Psi(\mathbf{x}_N) = 0 \end{aligned} \tag{2}$$

where

$$J(y) = \mathbf{x}_{e,N}^T S \mathbf{x}_{e,N} + \sum_{k=0}^{N-1} (\mathbf{x}_{e,k}^T Q \mathbf{x}_{e,k} + \mathbf{u}_k^T R \mathbf{u}_k) \tag{3}$$

with

$$\mathbf{x}_{e,k} = \mathbf{x}_k - \mathbf{x}_{d,k}. \tag{4}$$

This $\mathbf{x}_{d,k}$ generally can encode any desired state characteristics (for example, an optimal velocity), and is often used to specify a nominal desired trajectory.

The dynamics to be used for the trajectory generation is based on a snap input, the integrations of which give the trajectory and its derivatives up until the snap. If $q(t)$ (or q if the time index is dropped) represents the position values of the trajectory,

$$q(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}^T, \tag{5}$$

and dot notation is used to denote its derivatives, then the state vector of the dynamics, $\mathbf{x}$ is given as

$$\mathbf{x} = \begin{bmatrix} q \\ \dot{q} \\ \ddot{q} \\ q^{(3)} \end{bmatrix}. \tag{6}$$

The continuous time dynamics can then be given by the linear system

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{7}$$

where

$$A = \begin{bmatrix} 0_{2x2} & I_{2x2} & 0_{2x2} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & I_{2x2} & 0_{2x2} \\ 0_{2x2} & 0_{2x2} & 0_{2x2} & I_{2x2} \\ 0_{2x2} & 0_{2x2} & 0_{2x2} & 0_{2x2} \end{bmatrix} \tag{8}$$

and

$$B = \begin{bmatrix} 0_{2x2} \\ 0_{2x2} \\ 0_{2x2} \\ I_{2x2} \end{bmatrix}. \tag{9}$$

Where $\mathbf{u}$ is the snap input for the system, $\mathbf{u} = \begin{bmatrix} x^{(4)} & y^{(4)} \end{bmatrix}^T$. The input is piecewise continuous, but is not class $\mathcal{C}^0$ continuous. The piecewise continuity, however, does mean that the jerk is class $\mathcal{C}^0$ continuous, and the trajectory is class $\mathcal{C}^3$ continuous, due to the dynamics of the system. The need for $\mathcal{C}^3$ continuity instead of $\mathcal{C}^2$ is due to the fact that the zero-error epsilon-point control is to be used with this trajectory, which requires the third derivative of the trajectory.

For use in the discrete, multi-stage system optimization of Eq. 2, the exact discretization of these dynamics is used. In this case the discrete dynamics are given by

$$\mathbf{x}_{k+1} = \bar{A}\mathbf{x}_k + \bar{B}\mathbf{u}_k \tag{10}$$

where

$$\bar{A} = e^{A\Delta t} \tag{11}$$

and

$$\bar{B} = \int_0^{\Delta t} e^{A\tau} d\tau B \tag{12}$$

where Δt is the discretization step size and e^M is the matrix exponential of a matrix M.

The safe region constraints can be represented by linear inequality constraints for each time step from 0 to N. The constraints for each time step are the nominal constraints defined in relationship to the virtual leader, transformed and rotated according to the leader's position and orientation. If the nominal constraints and the leader trajectory are given as inputs, the constraints at each time step can be calculated, and only need be calculated once as part of initialization. Similarly, the desired position at a time instance $q_{d,k}$ can be calculated based on the desired offset vector from the leader, δ, and the leader's position and orientation. The calculation of the desired positions is given by

$$q_{d,k} = q_{l,k} + R(\psi_{l,k})\delta \tag{13}$$

where the subscript l refers to the leader, $\psi_{l,k}$ is the leader's orientation, and $R(\psi_{l,k})$ is a rotation matrix calculated from the leader's orientation by

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix}. \tag{14}$$

By following similar principles, the linear inequality constraints can be calculated for each time step. Given an A_q^L and b_q^L which are linear inequality constraints on position defined in terms of the leader's position (denoted by the superscripted L), an $A_{c,k}$ and b_k can be calculated for each corresponding time index k. For A_q^L and b_q^L the following inequality holds

$$A_q^L q^L \leq b_q^L \tag{15}$$

for q^L, a position defined with respect to, or in the frame of, the leader position. According to (13), the transform of a point in the leader frame to the inertial frame (I) is given by

$$q^I = q_l^I + R_L^I(\psi_l) q^L \tag{16}$$

where the time subscript k is dropped for conciseness. Solving for q^L yields

$$q^L = R_I^L(\psi_l) q^I - R_I^L(\psi_l) q_l^I. \tag{17}$$

This is then substituted into (15) to give the inequality

$$A_q^L \left(R_I^L(\psi_l) q^I - R_I^L(\psi_l) q_l^I \right) \leq b_q^L. \tag{18}$$

Distributing and rearranging gives

$$A_q^L R_I^L(\psi_l) q^I \leq b_q^L + A_q^L R_I^L(\psi_l) q_l^I \tag{19}$$

which is of the form

$$A_{q,k} q_k \leq b_k \tag{20}$$

where

$$A_{q,k} = A_q^L R^T(\psi_{l,k}) \tag{21}$$

$$b_k = b_q^L + A_{q,k} q_{l,k}. \tag{22}$$

Then $A_{c,k}$ is the concatenation of $A_{q,k}$ with zeros

$$A_{c,k} = \begin{bmatrix} A_{q,k} & 0_{px6} \end{bmatrix} \tag{23}$$

such that

$$A_{c,k} x_k \leq b_k. \tag{24}$$

These $A_{c,k}$ and b_k are concatenated to form the A_c and b of (2). This is done as follows

$$A_c = \begin{bmatrix} A_{c,0} & 0_{px2} & 0_{px8} & 0_{px2} & \dots & 0_{px8} & 0_{px2} \\ 0_{px8} & 0_{px2} & A_{c,1} & 0_{px2} & \dots & \vdots & \vdots \\ \vdots & & & \ddots & & \vdots & \vdots \\ 0_{px8} & 0_{px2} & \dots & \dots & \dots & A_{c,N} & 0_{px2} \end{bmatrix} \quad (25)$$

and

$$b = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix} \quad (26)$$

where each $A_{c,k} \in \mathbf{R}^{px8}$ and each $b_k \in \mathbf{R}^p$ with p being the number of constraints given by the Voronoi tessellation (which may be as little as $p = 1$ and as large as $p = n - 1$, where n is the number of vehicles in the formation).

III. Solution Outline

In formulating a solution method a few assumptions were made to outline a concrete version of the problem. First, the assumption is made that the input trajectory is used in creating the desired trajectory, $\mathbf{x}_d$ so that

$$\mathbf{x}_d = \begin{bmatrix} q_{in} \\ \dot{q}_{in} \\ 0_{2x1} \\ 0_{2x1} \end{bmatrix}. \quad (27)$$

The position is kept as straying too far from the nominal trajectory may result in breaking formation and other undesirable behavior. The velocity is kept as a placeholder in the case that there is a desired nominal velocity to attempt to maintain. The desired values for the other derivatives of q, however, are set to zero as minimizing these derivatives can help to improve smoothness of the trajectory.

Second, values for the cost matrices and a terminal constraint manifold are chosen. The terminal manifold, $\Psi(\mathbf{x}(t_f))$, is designed so that the ending position is incident on the desired final position

$$\Psi(\mathbf{x}_N) = q_{e,N}. \quad (28)$$

The instantaneous state cost matrix, Q, is a positive semi-definite diagonal matrix where the diagonal terms are

$$Q_{diag} = \begin{bmatrix} 1 & 1 & 0 & 0 & 10 & 10 & 10 & 10 \end{bmatrix}. \quad (29)$$

The instantaneous control cost matrix, R, is a positive definite matrix

$$R = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}. \quad (30)$$

The terminal state cost matrix, S, is a positive semi-definite diagonal matrix with diagonal terms

$$S_{diag} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (31)$$

This S matrix, however, does not have any effect given the choice of terminal constraint manifold. It is kept in the formulation as a placeholder so that if there are additional desired characteristics of the terminal state those can be addressed.

Finally, the linear inequality constraints for a time step are assumed to be just a bounding box about the point of the input trajectory at the same time step. These bounding box constraints are represented and transformed in the same way that the voronoi tesselation constraints would be represented and transformed. Future work will add in the constraints according to the Voronoi tessellation, but, for simplicity, these bounding box constraints are used to develop an initial solution.

Because the system is a linear system (resulting in linear dynamic constraints) with linear constraints and with a quadratic cost, a quadratic programming optimization solver called Operator Splitting Quadratic Programming (OSQP) is used as the underlying solver [6]. This allowed for a significant speed boost compared to non-linear solvers because of the ability to leverage problem and data structure. This did meant that the control portion of the solution is a custom implementation, but follows multiple-shooting concepts. Additionally, this required a slight reformulation of the cost to match the formulation of the solver. OSQP solves the problem

$$\begin{aligned} &\min_{\mathbf{x}} \ \frac{1}{2}\mathbf{x}^T P\mathbf{x} + \mathbf{q}^T\mathbf{x} \\ &s.t. \\ &\mathbf{l} \leq A\mathbf{x} \leq \mathbf{u} \end{aligned} \quad (32)$$

where $\mathbf{x}$ in this context would be analogous to $\mathbf{y}$ in the previous statement of the optimization problem, and A has no relation to the A of the dynamics, but is the linear inequality constraint matrix.

Thus the dynamic constraint must be reformulated as part of the linear inequality constraint, and the quadratic cost in terms of P and $\mathbf{q}$. By expanding terms in the cost $J(\mathbf{y})$ gives

$$\begin{aligned} J(\mathbf{y}) = &\mathbf{x}_N^T S\mathbf{x}_N - 2\mathbf{x}_{d,N}^T S\mathbf{x}_N + \mathbf{x}_{d,N}^T S\mathbf{x}_{d,N} \\ &+ \sum_{k=0}^{N-1} (\mathbf{x}_k^T Q\mathbf{x}_k - 2\mathbf{x}_{d,k}^T Q\mathbf{x}_k + \mathbf{x}_{d,k}^T Q\mathbf{x}_{d,k} + \mathbf{u}_k^T R\mathbf{u}_k). \end{aligned} \quad (33)$$

The minimization of this is equivalent to the minimization of the same cost without the terms that depend only on the desired values, $\bar{J}(\mathbf{y})$,

$$\begin{aligned} \bar{J}(\mathbf{y}) = &\mathbf{x}_N^T S\mathbf{x}_N - 2\mathbf{x}_{d,N}^T S\mathbf{x}_N \\ &+ \sum_{k=0}^{N-1} (\mathbf{x}_k^T Q\mathbf{x}_k - 2\mathbf{x}_{d,k}^T Q\mathbf{x}_k + \mathbf{u}_k^T R\mathbf{u}_k). \end{aligned} \quad (34)$$

This is equivalent to the form of the cost for OSQP with

$$P = \begin{bmatrix} P_{x,1} & 0 & \dots & \dots & 0 \\ 0 & P_{u,1} & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & P_{u,N-1} & 0 \\ 0 & \dots & \dots & 0 & P_{x,N} \end{bmatrix} \quad (35)$$

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_0 \\ 0 \\ \mathbf{q}_1 \\ 0 \\ \vdots \\ \mathbf{q}_N \end{bmatrix} \tag{36}$$

where

$$P_{x,k} = Q \qquad \forall k \in \{0, \ldots, N-1\} \tag{37}$$
$$P_{x,N} = S \tag{38}$$
$$P_{u,k} = R \tag{39}$$
$$\mathbf{q}_k = -2P_{x,k}\mathbf{x}_{d,k} \qquad \forall k \in \{0, \ldots, N\}. \tag{40}$$

The dynamics are converted to a set of linear equality constraints by relating a state to the previous state and input through the discrete dynamics. Encapsulating these relationships in a constraint matrix and vector gives

$$A_{eq}\mathbf{y} = \mathbf{b}_{eq} \tag{41}$$

where

$$A_{eq} = \begin{bmatrix} -I & 0 & \ldots & \ldots & \ldots & \ldots & 0 & 0 \\ \bar{A} & \bar{B} & -I & 0 & & & \vdots & \vdots \\ 0 & 0 & \bar{A} & \bar{B} & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & \ldots & 0 & \bar{A} & \bar{B} & -I & 0 \end{bmatrix} \tag{42}$$

and

$$\mathbf{b}_{eq} = \begin{bmatrix} -\mathbf{x}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{43}$$

with I being the identity matrix of appropriate dimension. This A_{eq} can be conjoined with the A_c matrix as defined previously to give a joint constraint matrix

$$A_{joint} = \begin{bmatrix} A_{eq} \\ A_c \end{bmatrix} \tag{44}$$

and joint lower and upper bound vectors defined by

$$\mathbf{l}_{joint} = \begin{bmatrix} \mathbf{b}_{eq} \\ -\infty \\ \vdots \\ -\infty \end{bmatrix} \tag{45}$$

and

$$\mathbf{u}_{joint} = \begin{bmatrix} \mathbf{b}_{eq} \\ b \end{bmatrix}. \tag{46}$$

In this way the problem is reformulated for using the OSQP solver to solve the parameter optimization problem resulting from the multiple shooting formulation of the discrete control problem.

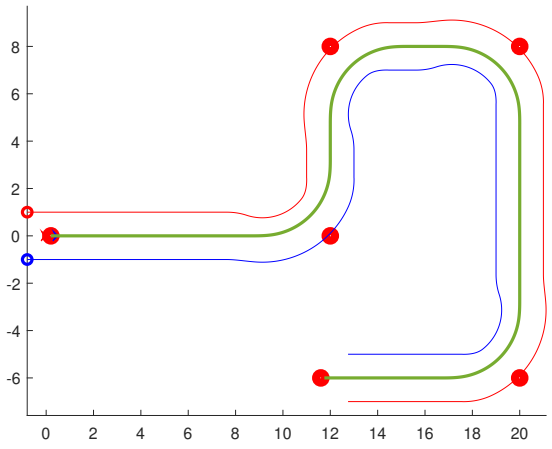

Fig. 1. A simplified version of a typical multi-vehicle formation scenario where path smoothing is helpful. A virtual leader trajectory (in green) is generated based on the red circular waypoints. The red and blue trajectories are follower trajectories generated based on the leader trajectory and a constant offset vector to each follower. Note that the follower trajectories have portions of the path that have undesirable and unnecessary displacements in position, even for this seemingly simple leader trajectory.

IV. Results and Analysis

In order to demonstrate the use cases for path smoothing, a simplified multi-vehicle formations control example scenario is used, shown in Fig. 1. In this scenario, several waypoints are given, and a trajectory for a virtual leader is generated using a form of Dubin's path generation with clothoid constant curvature modifications to achieve a path that is continuous in curvature that passes near each waypoint. Subsequently, follower agent trajectories are generated using the virtual leader trajectory and an offset vector for each follower agent to describe the desired offset from the virtual leader to the follower agent. This results in the trajectories shown in Fig. 1.

These derived follower agent paths have the advantage that an agent following its trajectory exactly is guaranteed to maintain the specified formation exactly. This also guarantees that a follower trajectory is always in the safe region of operation given by the Voronoi tessellation at every time. However, these follower trajectories can be less smooth than is desirable. In addition to the potentially undesirable displacements in position, there are also discontinuities in the linear and angular accelerations that contribute to difficulties in actuation and execution (discontinuities shown in Fig. 5). It is these difficulties that the optimal smoothing techniques presented herein attempt to mitigate.

Using the methods described in Section III, but without the box constraints at each time instance, gives good initial results. The resulting smoothed paths in comparison to the original paths are shown in Fig. 2. Figs.3, 4, and 5 show comparisons between the smoothed and the original trajectories in the curvature and the linear and angular velocities and accelerations of the trajectories.

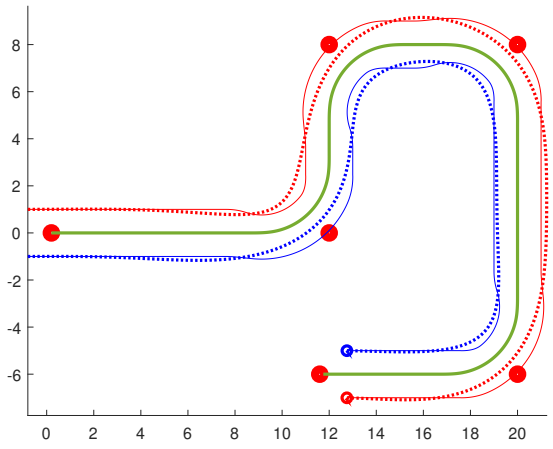

Fig. 2. A comparison of the smoothed paths without box constraints (dotted red and blue lines) to the original paths (solid red and blue lines). Even visually, these smoothed trajectories look more "smooth" and desirable.

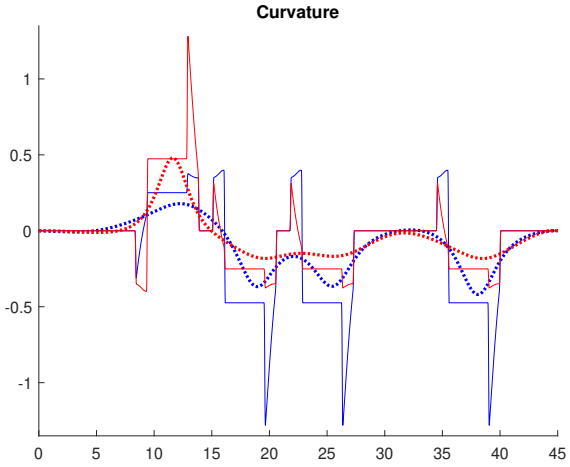

Fig. 3. A comparison of the curvature of the smoothed trajectories without box constraints (dotted red and blue lines) to the curvature of the original trajectories (solid red and blue lines). The original trajectory has instances of discontinuity in curvature, which the smoothed trajectory does not. Additionally, the magnitude of the curvature of the smoothed trajectories is typically much lower than the curvature of the original trajectories.

The original trajectories show discontinuities in all the states except for the linear velocity, while the smoothed trajectories are continuous in all of the states. In addition, it is of note that for all the states the magnitude of the value derived from the smoothed trajectory is typically much smaller than the value derived from the original trajectory. This is particularly true for the acceleration states. This is important as the control inputs for the vehicle dynamics considered for this project ([3]) are related to the accelerations. Because of this, the smoothed trajectories will result in a smaller magnitude of control authority utilized in actuation.

These trajectories, however, do have regions where the

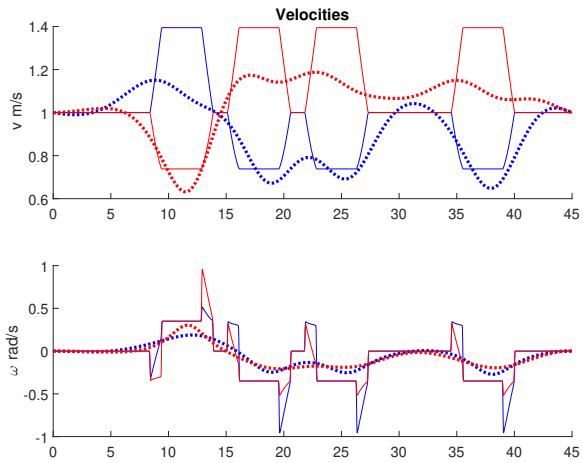

Fig. 4. A comparison of the linear (v) and angular (ω) velocities of the smoothed trajectories without box constraints (dotted red and blue lines) to the velocities of the original trajectories (solid red and blue lines). The original trajectory has instances of discontinuity in the angular velocity, but is continuous in the linear velocity. The smoothed trajectory, however, is continuous in both linear and angular velocity. Additionally, the magnitude of both velocities is less for the smoothed trajectories than the original trajectories (note that the linear velocity graph is centered on 1).

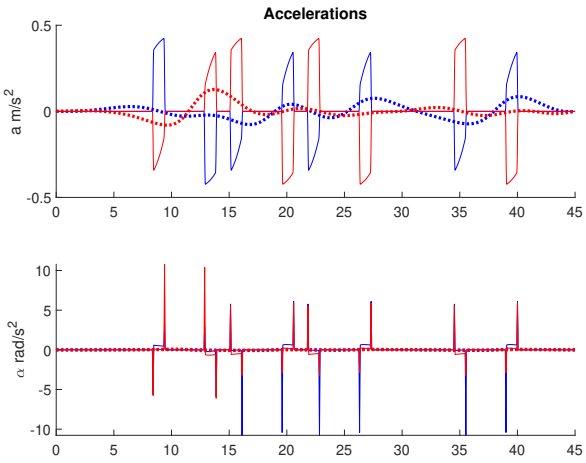

Fig. 5. A comparison of the linear (a) and angular (α) accelerations of the smoothed trajectories without box constraints (dotted red and blue lines) to the accelerations of the original trajectories (solid red and blue lines). The original trajectory is discontinuous in both linear and angular acceleration. The smoothed trajectory, however, is continuous in both linear and angular acceleration. Additionally, the magnitude of both linear and angular accelerations is less for the smoothed trajectories than the original trajectories, particularly so for the angular acceleration. These accelerations are indicative of the amount of control authority required to actuate and execute a trajectory, so the ability to reduce these values to this extent is quite desirable.

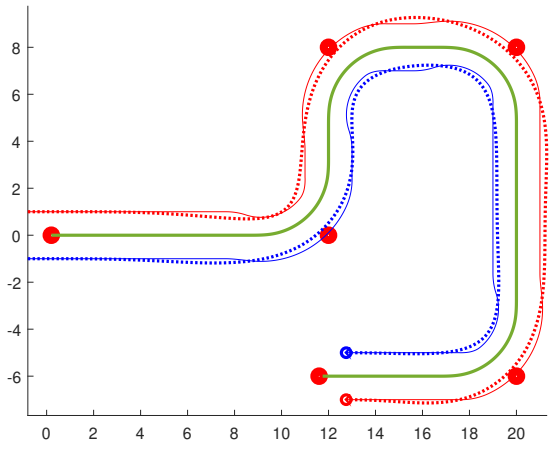

Fig. 6. A comparison of the smoothed paths with box constraints (dotted red and blue lines) to the original paths (solid red and blue lines). While less visually desirable than the unconstrained paths, the constrained smooth paths still appear better than the original paths

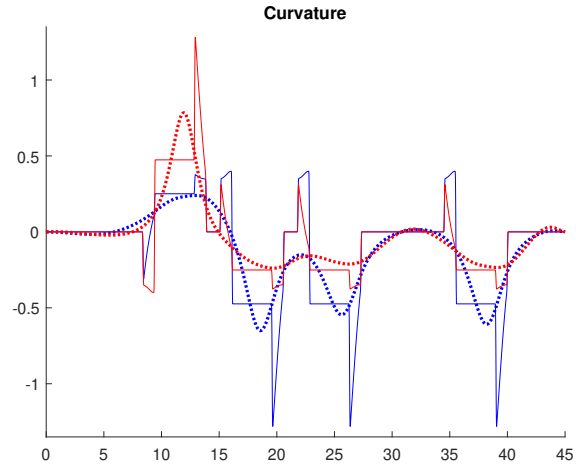

Fig. 7. A comparison of the curvature of the smoothed trajectories with box constraints (dotted red and blue lines) to the curvature of the original trajectories (solid red and blue lines). The constrained smoothed trajectory still has desirable continuity and magnitude properties although the magnitudes are less drastically reduced than in the unconstrained case.

position varies significantly from the original trajectory, which represents the desired location in the formation. This indicates the possibility of a smoothed trajectory that, if followed exactly, results in leaving a safe region of operation. While whether or not any violation occurs depends heavily on the exact formation definition (the desired offset of each follower agent from the virtual leader), the resulting Voronoi tessellation constraints, and the smoothed trajectory, there is no longer any guarantee of satisfying the constraints and no guarantee of avoiding catastrophic results. Indeed, if a square box constraint with a half meter side length is considered, the unconstrained smoothed trajectories of this example violate that constraint.

For this reason, the smoothing optimization is formulated to include these constraints. While the constraints from a Voronoi tessellation in general will not be box constraints, they will be linear inequality constraints that can be represented in the same form as a box constraint. It is possible, in fact, to have a Voronoi tessellation generate a box constraint for an agent if the formation is defined accordingly. This brief discussion validates the use of a box constraint as a simplified constraint to perform a first-pass analysis of the affect of the constraint on the optimization and the resulting trajectories.

Using the square box constraint with a half meter side length for the constraints on the smoothing optimization results in the trajectories and state values shown in Figs. 6, 7, 8, and 9.

The resulting paths from the constrained optimization can be seen to more closely follow the original trajectory, but still show visual "smoothness", albeit not as much as in the unconstrained case. It is no surprise, then, that the values of the curvature, velocities and accelerations are generally closer to the original trajectory for the constrained trajectories than for the unconstrained trajectories. The constrained trajectory state values, in general, are still lower in magnitude than those of the original trajectories, with the maximum magnitudes of the

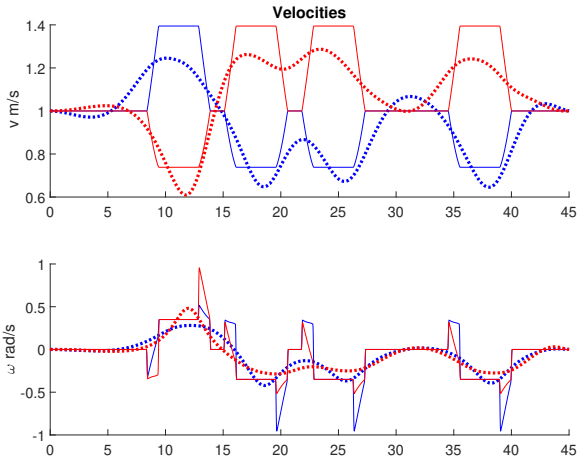

Fig. 8. A comparison of the linear (v) and angular (ω) velocities of the smoothed trajectories with box constraints (dotted red and blue lines) to the velocities of the original trajectories (solid red and blue lines). While the magnitudes of the velocities of the constrained smoothed trajectory are closer to the original trajectories than the unconstrained smoothed trajectories, they still represent an improvement in magnitude, and are still continuous.

values from the smoothed trajectories still being significantly smaller than those of the original trajectories. This is particularly true for the accelerations of the constrained smoothed trajectories that are still significantly smaller in magnitude than the accelerations of the original trajectories. As noted previously, the accelerations are what have the most impact on the amount of control authority required to execute a trajectory.

While this increase in the magnitudes of these state values compared to the unconstrained case is not ideal, it is the trade-off for the guarantee of a trajectory that does not violate the safe region constraints. While an optimization could be done

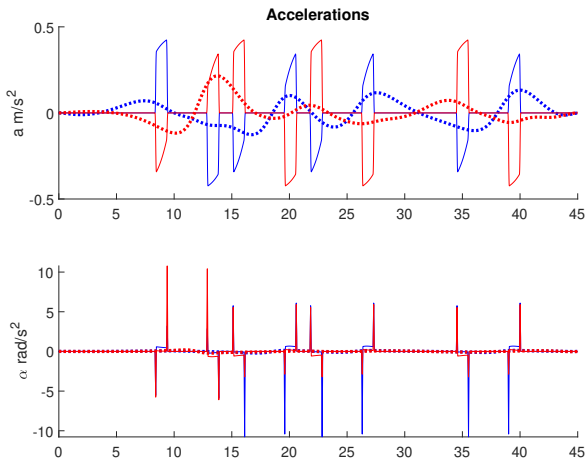

Fig. 9. A comparison of the linear (a) and angular (α) accelerations of the smoothed trajectories without box constraints (dotted red and blue lines) to the accelerations of the original trajectories (solid red and blue lines). Similar to the curvature and velocities, the accelerations from the constrained smoothed trajectory are not reduced in magnitude as much as in the unconstrained case, but are still an improvement over the original trajectories. Indeed, they are still typically a significant improvement.

TABLE I
OPTIMIZATION TIME FOR VARIOUS CASES

Optimization Case	Agent 1 Time (s)	Agent 2 Time (s)	Avg. Time
Unconstrained	0.328	0.315	**0.322**
Inactive Constraints	0.626	0.809	**0.718**
Active Constraints	1.949	2.358	**2.154**

Values for each case and agent are averaged over 5 runs each. Times obtained with an Intel i5-3470 3.20 GHz 4 core processor running Windows 10.

that considers all of the agents simultaneously, and therefore guarantees collision free trajectories, this quickly becomes intractable due to the "curse of dimensionality". Maintaining the safe region constraints guarantees collision free trajectories while also allowing the agents and their trajectories to be considered separately.

Another trade-off for having the constraints in place is also computation time. Table I shows the computation time of the OSQP optimization without constraints, with constraints that are large enough to not significantly affect the solution, and with constraints that significantly limit the possible solution. As can be seen, there is a significant difference in the computation time when the constraints are active, or make a significant difference in the final solution, versus when they are present, but not active. This indicates that there may be some cases when having the constraints may have a relatively low impact on computation time, and other cases when the constraints have a relatively large impact on computation time.

This increase in computation time, however, could potentially be mitigated through other means. For example, one could determine "key" points in the trajectory in which to apply the constraints. "Key" points could mean: periodically (e.g., once every second), during turns, when the curvature of the leader trajectory exceeds some threshold, or other points that are determined important for ensuring the trajectories are collision free. Additionally, parameter tuning for the OSQP algorithm could be done to decrease computation time, although this is typically a lengthy grid search process [7].

While there are trade-offs for constraining the optimization, there are cases where the benefits are worth the cost. The ability to guarantee collision free trajectories even though trajectories are considered independently is important in many applications, and can help to reduce online computations by limiting the frequency that agents need to avoid each other during execution. The continuity of the trajectory in curvature, velocities, and accelerations and the accompanying magnitude reduction are both beneficial for executability.

V. CONCLUSION

Planning for multiple vehicles in a formation requires the generation of collision free, executable trajectories for each agent. Considering optimal trajectories for all the agents together can generate smooth, collision free trajectories, but runs into the curse of dimensionality. Trajectories can be generated that are collision free and computationally efficient, but that often have regions of high vehicle accelerations that may not be executable, or that may be undesirable in other ways. By utilizing linear inequality constraints that allow for considering each vehicle separately while still guaranteeing collision free execution, an optimization framework is made that produces smooth and executable trajectories for each agent. While the smoothing optimization is more computationally expensive when considering these constraints, the benefit of guaranteeing an executable and collision free trajectory is often worth the extra computation time.

REFERENCES

[1] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777–790, 2001.

[2] C. Kownacki and L. Ambroziak, "Local and asymmetrical potential field approach to leader tracking problem in rigid formations of fixed-wing UAVs," *Aerospace Science and Technology*, vol. 68, pp. 465–474, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ast.2017.05.040

[3] B. Merrel and G. Droge, "Clothoid-based moving formation control using virtual structures," *International Conference on Intelligent Robots and Systems*, 2020.

[4] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C. C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors (Switzerland)*, vol. 18, 9 2018.

[5] M. Egerstedt and C. Martin, *Control theoretic splines : optimal control, statistics, and path planning*. Princeton University Press, 2010.

[6] B. Stellato and G. Banjac. (2019) Osqp 0.6.0 documentation. [Online]. Available: https://osqp.org/docs/index.html

[7] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2