Road-Following Formation Control of Autonomous Ground Vehicles

Masahiro Ono*, Greg Droge**, Havard Grip*, Olivier Toupet*, Chris Scrapper**, and Amir Rahmani*

***Abstract*— This work presents a novel cooperative path planning for formation keeping robots traversing along a road with obstacles and possible narrow passages. A unique challenge in this problem is a requirement for spatial *and* temporal coordination between vehicles while ensuring collision and obstacle avoidance. A two-step approach is used for fast real-time planning. The first step uses the A* search on a spatio-temporally extended graph to generate an obstacle-free path for the agent while the second step refines this path through local optimization to comply with dynamic and other vehicle constraints. This approach keeps robots close to their intended formation while giving them flexibility to negotiate narrow passages and obstacles, adhering to any given constraints.**

I. INTRODUCTION

The concept of formation control has received much attention in multi-vehicle systems as it requires tight coupling of vehicle motion while vehicles use local knowledge to achieve a collective outcome, e.g. [1, 2]. Much of the work has been on the development of local control laws which allow vehicles to move into static or moving rigid configurations in wide open areas, e.g. [1–5]. This paper presents a different type of formation control approach which is coupled to a tightly constrained route to be followed. Instead of rigid constraints with respect to the current position of other vehicles, spatial constraints are defined with respect to the progression along the road and the paths taken by other vehicles, much like lanes are spatially defined with respect to the center of the road. The approach has applications ranging from urban roads where lanes are well defined to offroad situations where a desired spatial constraint on the paths of the vehicles should be maintained.

The coordinated traversal of roads is not a new concept, work done on autonomous platooning dates back to the mid-nineties or earlier [6]. The idea behind platooning, however, is to define the longitudinal and lateral motions of the vehicles to move in a coordinated fashion along a well marked road [7]. Most techniques deal with the longitudinal spacing between vehicles moving in single file and require lanes or other reactive approaches to account for lateral spacing when the road is not straight [8]. They are not well suited for taking uncertainties such as ill-defined roads or obstacles into account.

Copyright 2015 California Institute of Technology and Space and Naval Warfare Systems Command (SPAWAR) Pacific. U.S. Government sponsorship acknowledged.

*M. Ono, H. Grip, A. Rahmani, and O. Toupet are with Robotic and Estimation group at Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109. {Masahiro.Ono, Havard.F.Grip, Olivier.Toupet, Amir.Rahmani}@jpl.nasa.gov

**G. Droge and C. Scrapper are with Space and Naval Warfare Systems Command, San Diego, CA 92152. {greg.droge, chris.scrapper}@navy.mil

Needs other than longitudinal spacing also arise for more complex formations. Take, for example, the three-vehicle snow removal team illustrated in Figure 1. The first two vehicles have snowplows and their objective is to clear a path as wide as possible, but maintain overlap in paths to ensure the passage is cleared. The third vehicle dispenses salt and must stay within the cleared path, while staying close enough to the front vehicles. The vehicles have to flexibly adapt the formation in order to safely follow the road. For example, they may have to deviate from the ideal formation in order to avoid obstacles or to go through a narrow passage.

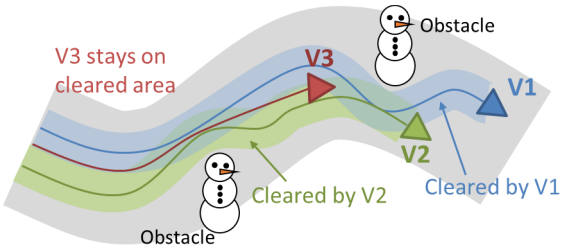


Fig. 1: A snowplowing scenario where two front vehicles clear the road while the third vehicle stays in the cleared path and dispenses salt.

Such a road-following formation control problem poses three challenges: a) formation control with obstacle and collision avoidance, b) tractable solving of non-convex optimization, and c) handling of spatially and temporally coordinated formations. Details of these challenges as well as our approach to overcome them are described below.

A. Formation control with obstacle and collision avoidance

The vehicles must intelligently balance adherence to a specified formation with safety (ie obstacle and inter-vehicle collision avoidance). The majority of existing research on formation control have focused on the convergence to the desired formation, while deliberative deviation from it is often not considered. We employ a receding horizon control (RHC)-based formation control approach [9], which enables deliberative formation control and obstacle/collision avoidance by solving a constrained optimization problem in real-time.

B. Tractable non-convex optimization

The constrained optimization problem to be solved in real-time includes obstacle and collision avoidance constraints which make the problem non-convex. These non-convexities combined with vehicle dynamic considerations raise a serious computational challenge.

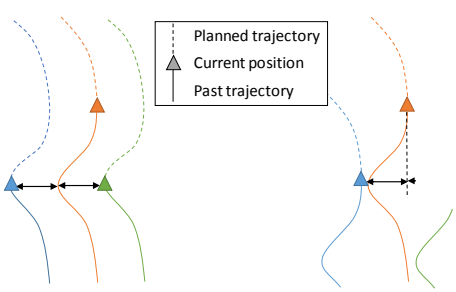
While global optimization techniques exist, there is difficulty in keeping the problem tractable. For example, an existing approach to globally optimize a similar problem is to encode it as a mixed-integer programming problem where obstacles are represented as arithmetic inequality constraints, e.g. [10]. However, such a representation for obstacles is not directly applicable to the more common occupancy grid approach and the computation grows exponentially with the number of obstacles.

A more tractable approach is to use local optimization techniques as in [11]. While such a gradient-based method can directly handle occupancy maps and converges very quickly, it is sensitive to the initial solution. As in [11], there are often techniques for introducing smarter initial conditions. However, these techniques may not always be extensible to the high dimensional state space of a multi-vehicle system.

We marry solution efficiency and tractability by employing a two-layer architecture, consisting of a graph-based planner and a local optimizer. First, a graph planner quickly generates an obstacle-free path using an A* search on a discretized state lattice with a simplified vehicle dynamics. This path is then used as an initial solution for a local optimizer, which refines the paths. Although this approach does not provide a guarantee of global optimality, practically it is capable of generating a near-optimal path very quickly.

C. Spatially- and temporally-coordinated for

Road-following formation control consist
ceptually distinct types of coordination, whic
tial coordination and *temporal coordination*.
coordinated formation, the trajectories of tl
spatially aligned, as in Figure 2-(a). Note t
positions of the vehicles at a given point i
coordinated. In a *temporally-coordinated for*
ative positions of the vehicles at the same



(a) Spatially-coordinated formation (b) Temporally-coordinated formation

Fig. 2: Examples of spatially- and temporally-coordinated formations.

A practical road-following formation control problem is usually a hybrid of the two. For example, purely spatial coordination does not prohibit a follower vehicle to go ahead of the leader. To prohibit such a behavior, we need to impose a temporal coordination constraint that, at any given point of time, the positions of the followers are behind the leader.

Planning with spatio-temporal constraints has been studied in the AI planning community, represented by a classical example of STRIPS (Stanford Research Institute Problem Solver) [12] and a more recent example of PDDL (Planning Domain Definition Language)-based planners [13]. However, their capability to handle a problem with continuous dynamics such as path planning is limited.

Existing research in formation control has mostly focused on the temporal coordination. A main contribution of this work is a framework that can handle a hybrid of spatially and temporally coordinated formations.

The rest of the paper is organized as follows: In Section II we formally define key concepts, such as spatial and temporal coordination constraints, and formulate the problem. Section III presents our solution approach, which consists of a graph planner and a local path optimizer. Finally Section IV presents simulation results.

II. PROBLEM FORMULATION

A. Coordinate System

We first define the sr coordinate system, which we use for planning. As shown in Figure 3, the s axis is laid along the road, typically corresponding to the center of the road.

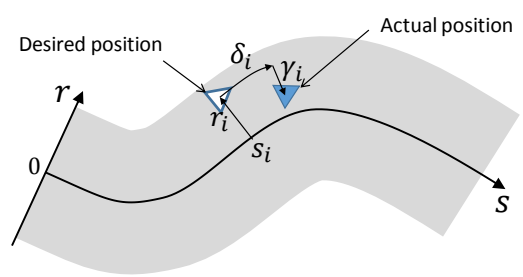


Fig. 3: The s-r coodinate system that we use for planning.

B. Definition of Desired Vehicle Position

We separate the planning problem into two parts. The first part is to plan the desired position for each vehicle, while the second part is to control each vehicle to converge to the desired position while avoiding obstacles and satisfying various constraints. The first part is described in this subsection while the second part is described in Section II-C.

We specify the desired position of each vehicle in the sr coordinate, as shown in Figure 3. We first specify s_i, the s coordinate for the i'th vehicle, as a function of time, denoted by $s_i(t)$. An underlying assumption is that s_i is monotonically increasing with t, meaning that vehicles do not back up. This limitation can be removed but it is beyond the scope of this paper. Then we specify r_i for a given s,

denoted by $r_i(s)$. Spatial and Temporal coordination can be realized by imposing the following constraints on s_i and r_i.

- **Spatial coordination** between vehicle i and j is realized by specifying the lateral offset, c_{ij}^S, between the two vehicle at given s, as follows:

$$r_i(s) - r_j(s) = c_{ij}^S, \quad \forall s \geq 0. \tag{1}$$

- **Temporal coordination** between vehicle i and j is realized by specifying the lateral offset, c_{ij}^T, between the two vehicle at given t, as follows:

$$r_i(s_i(t)) - r_j(s_j(t)) = c_{ij}^T, \quad \forall t \geq t_0. \tag{2}$$

C. Path Planning

We allow vehicles to deviate from the desired position in order to avoid obstacles and satisfy various coordination constraints. As shown in Figure 3, the deviation of the i'th vehicle is represented in the sr coordinate by (δ_i, γ_i). We parameterize δ_i and γ_i by s_i. This means that, when a given desired position is at $(s_i, r_i(s_i))$, the deviation from it is given by $(\delta_i(s_i), \gamma_i(s_i))$.

The control objective is to minimize the deviation. Letting S be a finite set of s over a planning horizon, the objective function is:

$$\min \sum_{s \in S} ||(\delta_i(s), \gamma_i(s))||.$$

Note that the above objective function can also be parameterized by t by replacing s with $s_i(t)$.

We also impose the following constraints:

- **Obstacle avoidance constraint** is imposed on each vehicle as follows:

$$(s + \delta_i(s),\ r_i(s) + \gamma_i(s)) \in \mathcal{R},$$

where $\mathcal{R}$ is the obstacle-free space.

- **Spatial coordination constraint** betwee j is generally represented as follows:

$$f_S(r_i(s) + \gamma_i(s),\ r_j(s) + \gamma_j(s)) \leq 0,$$

where f_S is a real-valued function.

- **Temporal coordination constraint** betv and j is generally represented as follow

$$f_T(s_i(t) + \delta_i(s_i(t)),\ s_j(t) + \delta_j(s_j(t)),$$
$$r_i(s_i(t)) + \gamma_i(s_i(t)),\ r_j(s_j(t)) +$$

where f_T is a real-valued function.

We note that the spatial and temporal be imposed between any two vehicles, reg dependency tree. By specifying the constrai nated path planning problem is formulated a optimization problem, where vehicles are c spatial and temporal coordination constraints. tion problem is solved by a combination o and local optimization, which will be explai Section III.

Examples: An example of spatial coordination constraint is to require that Vehicle 2 is always on the left of Vehicle 1's trajectory. Such a constraint is prescribed as:

$$(r_1(s) + \gamma_1(s)) - (r_j(s) + \gamma_j(s)) \geq 0, \quad \forall s \geq 0.$$

An example of temporal coordination constraint is collision avoidance, which requires that the distance between Vehicles 1 and 2 is always greater than d. Such a constraint is prescribed as:

$$(s_i(t) + \delta_i(s_i(t)) - s_j(t) - \delta_j(s_j(t)))^2$$
$$+ (r_i(s_i(t)) + \gamma_i(s_i(t)) - r_j(s_j(t)) - \gamma_j(s_j(t)))^2 \geq d^2,$$
$$\forall t \geq t_0.$$

Practical applications usually have both types of constraints.

III. COOPERATIVE PATH PLANNING

A. Receding Horizon Planning

We employ a receding horizon planning approach, where the planner plans over a finite horizon and only the first control input is executed for each vehicle, as illustrated in Figure 4. More specifically, planning is performed by repeating the followings for each vehicle at every planing cycle:

1) Specify the desired position relative to the reference vehicle
2) Plan an obstacle-free path over the planning horizon using the graph planner
3) Use the path from the graph planner as an initial solution, plan a locally-optimal, dynamically-feasible

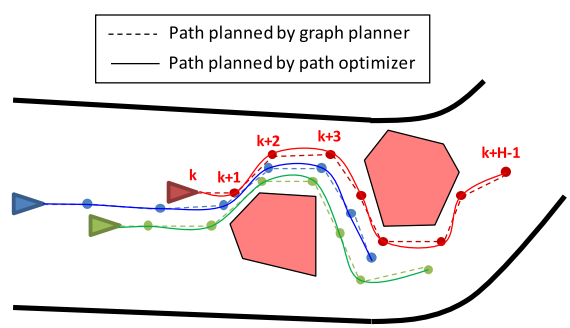


Fig. 4: Illustration of Receding horizon control.

B. Sequentially Distributed, Asynchronous Planning

In our approach, computation is distributed over vehicles sequentially. We define a dependency tree, where all the vehicle except the leader has a *reference vehicle*. For this work we assume that the dependency tree is acyclic and built in a centralized manner. Each vehicle computes its own plan asynchronously. More specifically, when each vehicle finishes computing a plan, it publishes the plan to the followers; the followers plan by using the latest plan of the reference

vehicle as an input. In this way, even though the in flow is sequential, the computation does not ne sequential. While such a sequentially distributed approach does not result in a globally optimal s general, it has a practical advantage in that it instability issues due to the existence of feedback l Moreover, unlike formation control of homogeneou in a free space, in many practical road-following control applications there is an obvious dependency between vehicles, as in the snowplow scenario de Section I.

The path of the leader vehicle is obtained b a path planning problem without coordination c The desired position of all other vehicles are ob propagating (1) or (2) down the dependency tree. L j be the reference vehicle of Vehicle i. Given the planned path of Vehicle j as $r_j(s_j(t))$, the desired p Vehicle i is computed from (1) or (2). Note that the leader vehicle can be replaced by a *virtual leader*, an imaginary vehicle that works as a reference point (such as the center of the road, i.e., $r_{VL} = 0, \forall s \geq 0$).

C. Graph Planner

Our approach to handle spatial and temporal coordination constraints is to perform search on a spatio-temporally extended graph for each vehicle. Intuitively, a spatio-temporally extended graph is a discrete representation of spacetime. As shown in Figure 5, it consists of nodes on multiple "slices" of time, where each slice is a two dimensional space. The flow of time is unidirectional. Hence, nodes on slices at adjacent time steps are connected by uni-directional edges pointing from the past slice to the future slice. Nodes on the same slice are not connected because a vehicle cannot jump to a different location momentarily. On each slice, nodes are populated in the δ_i-γ_i space, or in other words, the deviation

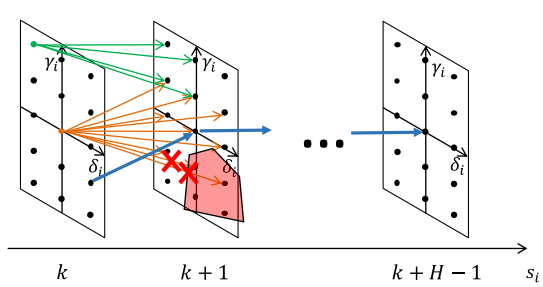


Fig. 5: Spatio-temporally extended graph used for path planning.

Details are now presented on the construction and search of the graph:

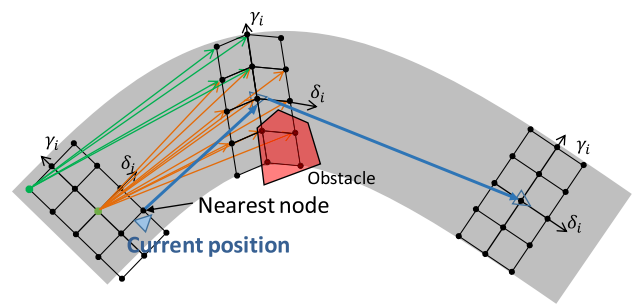


Fig. 6: Mapping of the spatio-temporally extended graph in Figure 5 to the s-r coordinate.

1) Construction of a graph: The construction of a spatio-temporally extended graph typically requires non-trivial computation time due to its relatively large number of nodes. Our approach is to shift most of the computation off-line by employing a decremental graph construction method. More specifically, a graph that connects all approximately dynamically feasible edges *without* considering constraints is constructed *a priori*. This unconstrained graph is invariant in time. Then, in real time, at each time step, the planner checks constraints for each node, and removes edges that are connected to nodes violating a constraint.

The off-line, unconstrained graph for the i'th vehicle is a tuple $< V_i, E_i >$, where V_i and E_i are the set of nodes and edges, respectively. Let S_i be the set of s_i in the planning horizon. For each $s_i \in S_i$, we populate a finite number of points in a rectangular lattice in the δ-γ space, as in Figure 5. In the δ direction nodes are populated within $[\delta_{\min}, \delta_{\max}]$ at a constant interval, where $\delta_{\min}$ and $\delta_{\max}$ are given parameters. We denote by Δ the set of all the δ of the nodes. In the γ direction nodes are populated so that they cover the maximum width of the road. We denote by Γ the set of all the γ of the nodes. The set of all nodes in the spatio-temporal graph is given by:

$$V_i := S_i \times \Delta \times \Gamma.$$

Unidirectional edges connect two nodes at adjacent slides, pointing forward in time. Ideally E_i should include only dynamically feasible transitions. However, since the graph represents only a subset of the vehicle's state (e.g., it does not include velocity), it cannot fully account for vehicle dynamics. As a result, the graph planner uses approximate dynamical constraints, where a transition between nodes is allowed only when its magnitude in the $\delta\gamma$ space is below a threshold:

$$\begin{aligned} &[(s_i^k, \delta^k, \gamma^k) \rightarrow (s_i^l, \delta^l, \gamma^l)] \in E_i \\ &\text{iff} \quad s_i^l - s_i^k = \Delta s_i \quad \wedge \quad ||(\delta^k - \delta^l, \gamma^k - \gamma^l)|| \leq c, \end{aligned}$$

where $[(s_i^k, \delta^k, \gamma^k) \rightarrow (s_i^l, \delta^l, \gamma^l)]$ is the edge from node $(s_i^k, \delta^k, \gamma^k)$ to $(s_i^k, \delta^k, \gamma^k)$, Δs_i is the interval of s_i in S_i and c is a threshold. The detailed vehicle dynamics is taken into account by the local path optimizer. The above computation is performed off-line, and the resulting graph is stored in each vehicle's memory.

On-line computation is required to evaluate obstacle avoidance, spatial coordination, and temporal coordination constraints. For each nodes $v \in V^i$, the planner checks the satisfaction of the constraints (3)-(5). If one or more constraints are violated at a node, all the edges connected to the nodes are removed from the graph. The obstacle avoidance constraints (3) are check at all the nodes on all the slices in the graph. We note that the spatio-temporally extended graph allows for the handling of moving obstacles.

The spatio-temporal graph representation fully accounts for both spatial and temporal coordination constraints, (4) and (5). Let s_i^k and t_i^k be the s coordinate and the time at the kth slice of the spatio-temporal graph. (Hence $s_i(t_i^k) = s_i^k$.) Assuming that Vehicle j is the reference vehicle of Vehicle i, let $r_j(s_j(t))$ be the past and planned path of Vehicle j. Then, for all nodes on the kth slice of the graph, the spatial coordination constraint (4) is checked against $r_j(s_j^k)$, while the temporal coordination constraint (5) is checked against $(s_j(s_j(t_i^k), r_j(s_j(t_i^k))))$. Note that $s_i^k \neq s_j(t_i^k)$.

2) Search: The A* algorithm is used to find the optimal path on the spatio-temporally extended graph. The initial node is the one that is closest to the current position of the vehicle at the first slice of the graph; the search terminates once it reaches one of the nodes in the last slice of the graph.

The control objective is to keep vehicles as close as possible to the desired position. Therefore the planner penalizes the aggregated deviation from $(\delta_i, \gamma_i) = 0$ over the planning horizon. In a typical formulation of A*, cost is associated to edges; we use a variant cost is associated with nodes. More specifically, we use the following cost function:

$$\sum_{k=1}^{H} \left(\delta_i (s_i^k)^2 + \gamma_i (s_i^k)^2\right),$$

where H is the number of time steps in the horizon.

In order to guarantee the optimality of the A* search, heuristics must be admissible, meaning that it must not overestimate the cost-to-go from every node. For the leader vehicle we use the cost of unconstrained optimal path from each node as the heuristic. This heuristic is pre-computed by optimizing the path without constraints from each node. It is easy to show that the cost of unconstrained optimal path always underestimate the cost of constrained optimal path. Since the leader vehicle has already found an obstacle-free path and the desired position of the followers are specified relative to the leader, the solution typically lies very close to the desired position. Hence, for the follower vehicles, we do not use heuristics, but instead use a greedy search to find the optimal solution very quickly.

D. Local Path Optimizer

The output of the graph planner is a series of discrete points which take into account the spatial and temporal constraints imposed on the system. The nature of the graph-based search forces the chosen parameters to be conservative as the state of the vehicle is abstracted to allow for a low dimensional search for the optimization. The purpose of the local planner is to generate a continuous path which satisfies both the temporal and spatial constraints imposed on the vehicles while considering path smoothness and vehicle parameters along the path, such as vehicle dimensions, orientation, velocities, and accelerations.

To make a continous transition from the output of the graph planner to continuous space, we utilize a control theoretic spline, [15], for each of the sr variables. Such splines allow for a perfect interpolation of the input points with the least amount of curvature. The points can then be adjusted, as shown in Figure 7, using an optimal control framework to optimize the desired parameters. As the splines are semi-parametric, the optimization of the trajectory is greatly simplified as it can be done using parameter optimization techniques.

We now briefly describe the concept of control theoretic splines, define the splines for the sr coordinates, and formulate the optimal control problem.

1) Control Theoretic Splines Background: The notation and development in this section follows closely the work in [15]. Control theoretic splines are defined in terms of the output, $y \in \mathbb{R}$, of a linear system with state $x \in \mathbb{R}^n$ and input $u \in \mathbb{R}$. The differential equations defining the relationship between these variables can be expressed as

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}, \tag{6}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in R^{n\times 1}$, and $C \in \mathbb{R}^{1\times n}$.

The fundamental idea behind control theoretic splines is that the solution for y can be expressed in terms of a basis function, $l_t(s)$, as:

$$y(t) = Ce^{At}x_0 + \int_0^t l_t(s)u(s)ds. \tag{7}$$

where the basis function is parameterized by t and is a function of s. It is expressed in terms of the linear system variables as:

$$l_t(s) = \begin{cases} Ce^{A(t-s)}B & t > s \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

The spline can then be formulated by defining the input as a weighted summation over a number of basis functions with differing values of t. Specifically, defining a number of points in time, $t_1, t_2, \ldots, t_N$, as well as the same number of scaling values, $\tau_1, \tau_2, \ldots, \tau_N$ the input can be written as

$$u(s) = \sum_{i=1}^{N} \tau_i l_{t_i}(s). \tag{9}$$

Finally, given a vector of points, $\alpha \in \mathbb{R}^N$, to interpolate between, i.e. $y(t_i) = \alpha_i$, the scaling factors, $\tau = [\tau_1, ...\tau_N]^T$, can be defined to achieve the least amount of control effort, this corresponds to minimizing curvature as follows:

$$\tau = \alpha^T G^{-1},\ G = \int_0^T l(s)l(s)^T ds \tag{10}$$

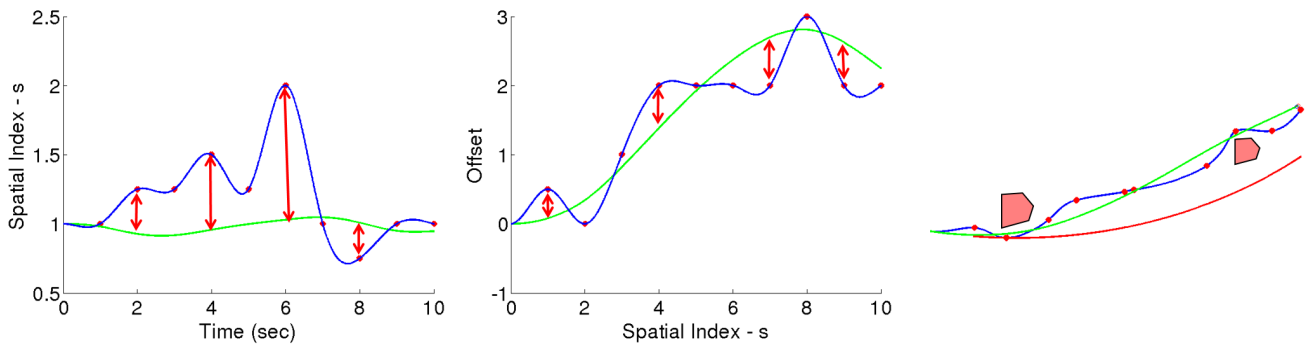


Fig. 7: This figure shows the result of using the optimization to smooth the inputs given by the graph planner. The arrows illustrate the movement of a few of the points. The far right image shows the resulting trajectory.

where $l(s) = [l_{t_1}(s), l_{t_2}(s), ..., l_{t_N}(s)]^T$. Note that G^{-1} exists assuming $t_i \neq t_j$ for $i \neq j$. Also, as long as the number of points being interpolated and the spacing between them stays the same, G^{-1} need only be calculate one time at initialization. Equation (10) is used to interpolate both the inputs from the graph planner and then the values for α are optimized[1] through optimal control based parameter optimization.

2) Defining Splines for the sr Variables: A separate spline is defined for both s and r with the subscripts a_s and a_r denoting variable a for the s and r spline respectively. To naturally minimize the curvature of the path, the linear systems are designed as second order systems so that the minimization of the input corresponds to minimizing the curvature of each spline. To allow the parameters to take on specific roles in the optimization, we allow s to be a function of time and r to be a function of s. This allows the offset r to shape the path and the spatial index s to shape the velocities. The dynamics for the systems can thus be described as[2]:

$$\begin{aligned} \frac{dx_s}{dt}(t) &= Ax_s(t) + Bu_s(t), & \frac{dx_r}{ds}(s) &= Ax_r(s) + Bu_r(s), \\ s(t) &= \begin{bmatrix}1 & 0\end{bmatrix} x_s(t) & r(s) &= \begin{bmatrix}1 & 0\end{bmatrix} x_r(s) \end{aligned} \tag{11}$$

where

$$A = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}, B = \begin{bmatrix}0\\ 1\end{bmatrix}. \tag{12}$$

Also note that $\frac{ds}{dt} = Dx_s$ and $\frac{dr}{ds} = Dx_r$ where $D = \begin{bmatrix}0 & 1\end{bmatrix}$.

Allowing the reference position (whether it be another vehicle or a virtual vehicle) be denoted as $q_l(s)$, the orthogonal and tangential unit vectors for motion be written as $h_O(s)$ and $h_T(s)$, and the relations $\frac{dh_O}{ds}(s) = -\omega(s)h_T(s)$ and $\frac{dh_T}{ds}(s) = \omega h_O(s)$, the desired position and its derivatives can be expressed in cartesian coordinates as:

$$\begin{aligned} q_d(t) =& q_l(s) + r(s)h_O(s) \\ \frac{dq_d}{dt}(t) =& \Big(\frac{dq_l}{ds}(s) + Dx_r(s)h_O(s) \\ & - \omega(s)r(s)h_T(s)\Big)Dx_s(t) \\ \frac{d^2q_d}{dt^2}(t) =& \Big(\frac{d^2q_l}{ds^2}(s) + u_r(s)h_O(s) - 2\omega(s)Dx_r(s)h_T(s) - \\ & r(s)\alpha(s)h_T(s) - r(s)\omega^2(s)h_O(s)\Big)(Dx_s(t))^2 \\ & + \Big(\frac{dq_l}{ds}(s) + Dx_r(s)h_O(s) \\ & - \omega(s)r(s)h_T(s)\Big)u_s(t) \end{aligned} \tag{13}$$

where $\alpha(s) = \frac{d\omega}{ds}(s)$, and for readability we have denoted $s(t)$ as s.

3) Optimal Control Problem: By maintaining the notation in terms of the sr coordinate system, the spatial and temporal constraints can be expressed naturally. As a matter of implementation the spatial constraints correspond to upper and lower bounds on the optimization. Moreover, as the splines are semi-parametric, the continuous path can be optimized through the optimization of a number of parameters and the spatial constraints as upper and lower bounds.

As the cost is the means of collaboration with the other vehicles, a subscript i denotes the vehicle performing the optimization and subscript j denotes the variables from another vehicle. Also, for shorthand we define $x_i(t) = \begin{bmatrix}x_{s_i}(t) & x_{r_i}(s_i(t))\end{bmatrix}$ and $\bar{q}_i = \begin{bmatrix}q_{d_i}(t) & \frac{dq_{d_i}}{dt}(t) & \frac{d^2q_{d_i}}{dt^2}(t)\end{bmatrix}$ The following generalized form of the cost can be used for the optimization of the continuous path:

$$\min_{\alpha_s, \alpha_r} \int_{t_0}^{t_f} \Big(L(x_i(t), q_i(t), s_j(t), r_j(t)\Big)dt + \Phi(x_i(t_f)) \tag{14}$$

s.t. Spatial and temporal constraints

Spline dynamics: (11)

where again α_s and α_r are the desired values of the spline, as defined generally in (10).

Two notes are in order on aspects of implementation

[1]Note that optimizing over the desired values, α, instead of the scaling parameters, τ, is a design choice. It is done as the spatial and temporal constraints can be expressed naturally over the desired values and in many cases simply be upper and lower bounds.

[2]Note that for sake of brevity, we do not give an analytic solution, but (7)-(10) and (12) can be combined to form an analytic solution to avoid the need for integration.

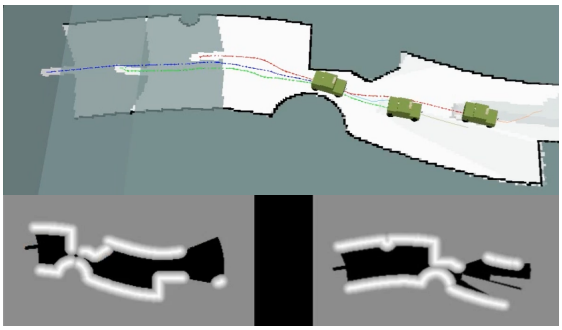


Fig. 8: The simulated environment including the paths taken by the vehicles (top); Each vehicle generates and uses its own costmap using simulated laser rangefinders (bottom).

1) The values of α_s and α_r can be initialize using the values passed in from the graph planner.
2) To maintain consistency with the route that the graph planner takes through obstacles, a terminal constraint can be added to not allow the final α values to deviate much from the input (or simply fix them to not be included in the optimization).

IV. Simulation Results

In order to validate the proposed approach, it was implemented in simulation and demonstrated in a multi-vehicle road-following scenario.

A. Implementation

The graph planner and the local path optimizer were implemented in C++, building upon the Robot Operating System (ROS) framework. The planner for each vehicle is run as a separate process (i.e., ROS node) but on the same machine. The planners were integrated with the ROS Stage simulator, an off-the-shelf mobile robot simulator, which simulates vehicle kinematics. The integrated simulation was run on a machine with Intel Xeon CPU clocked at 3.10 GHz and 16 GB RAM.

B. Simulation Settings

The simulation is designed in a way that can validate that the proposed road-following formation control approach overcomes the three challenges identified in Section I. We use the snowplow scenario presented in Section I. As shown in Figure 8, we consider a formation with three vehicles traversing a unmarked road with obstacles. Their desired positions are specified both spatially and temporally. In the lateral direction, the trajectory of Vehicle 2 should be 2.0 meters to the right of Vehicle 1; the trajectory of Vehicle 3 should be 1.0 meters to the left of Vehicle 2. These are spatial coordination requirement represented by (1). In the longitudinal direction, the desired position of Vehicle 2 is 10 meters behind Vehicle 1, while that of Vehicle 3 is 10 meters behind Vehicle 2. These are temporal coordination requirement represented by (2).

Obstacles were placed so the vehicles cannot stay in the desired formation in order to avoid them. Therefore, the expected behavior of the vehicles is to maintain the formation as much as possible but deviate from it in order to pass the obstacles.

Recall that planners essentially solve a constrained optimization problem. The following spatial and temporal constraints are imposed:

- (Spatial) Vehicle 2 must stay within 2.5 meter from the trajectory of Vehicle 1, in order to maintain overlap in paths to ensure the passage is cleared
- (Spatial) Vehicle 3 must be between the trajectory of Vehicles 1 and 2 to stay within the cleared area
- (Temporal) Vehicles must maintain at least 5.0 meter trailing distance in order to avoid collision

The length of the planning horizon is set to 15 steps. For each time step, the graph planner populates 87 nodes. Therefore, the spatio-temporally extended graph has 1305 nodes in total. The local optimizer has two variables to optimize for each step, which gives a total of 30 variables.

We evaluate the simulation results in two metrics, that are representative about how well the desired formation is achieved in terms of spatial and temporal coordination. One is the lateral distance between the trajectories of Vehicles 1 and 2, while the other is the trailing distance between Vehicles 1 and 2, and 2 and 3.

C. Simulation Result

Figure 8 is a snapshot from the simulation, showing that the three vehicles successfully maintain the desired formation whenever it is possible while deviating from it in order to avoid obstacles. Figure 9 shows the profile of the two metrics defined above. The width of cleared path stayed near the desired value, 2.0 meters, except between $s = 35 - 55$ and $s = 70-80$ when vehicles avoided obstacles. Also note that it was always below 2.5 meter, indicating that the constraint on the maximum lateral spread was always satisfied. The trailing distance between vehicles stayed almost at the desired level (20 meters between Vehicle 1 and 3, 10 meters between Vehicle 2 and 3). The dip observed around $s = 80$ was because Vehicle 1 reached the end of the road. The constraint on the minimum trailing distance between vehicles were always satisfied. The average computation time per vehicle per planning cycle was 0.0016 sec for the graph planner, and 0.69 sec for the local path optimizer.

Figure 10 shows the behavior of three vehicles in a triangular formation going through a narrow passage. Observe that the second vehicle waits before the passage to make way for the first vehicle. The third vehicle also slows down to avoid collision with the second vehicle. After the vehicles pass the passage, the formation is quickly restored. Such an intelligent collective behavior is a result of the fact that vehicles respect the temporal constraint to avoid collisions as well as the spacial specification of the formation.

V. Concluding Remarks

A traceable and efficient trajectory generation algorithm for spatially and temporally constrained robot formations traveling along a road is presented.

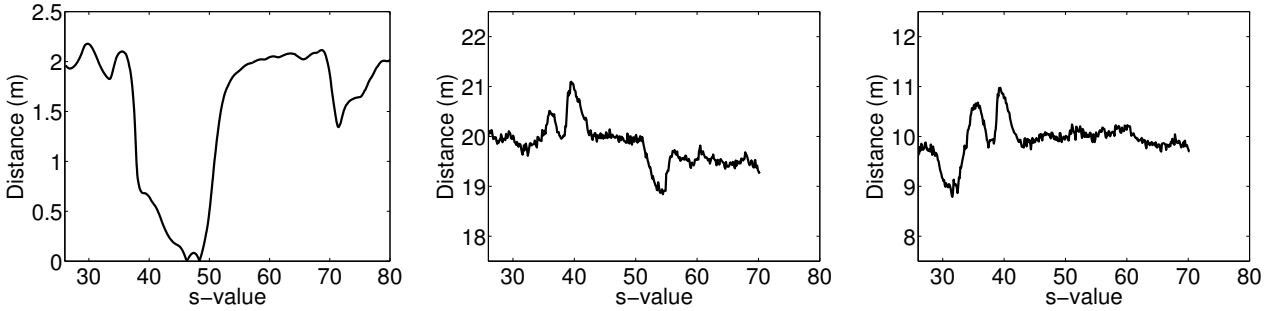


Fig. 9: This figure shows various metrics from the simulation shown in Figure 8. From left to right: width of cleared path, distance from last vehicle to first vehicle, and distance from last vehicle to second vehicle. The x-axis in each plot consists of the spatial distance, s, along the path.

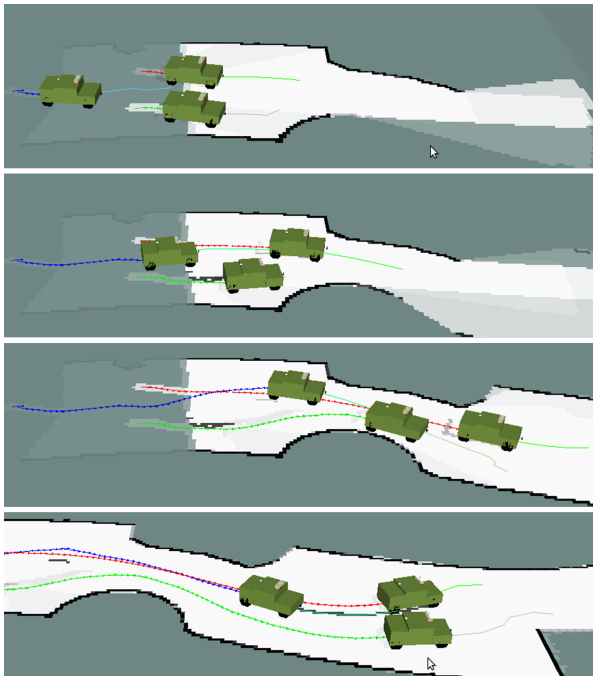


Fig. 10: Three vehicles in a triangular formation go through a narrow passage.

Simulations in a ROS based multi-robot environment with accurate kinematics and simulated sensor readings prove the viability of this method. We plan on real-world hardware deployment of this path planning approach.

VI. ACKNOWLEDGMENTS

Work supported by the Office of Naval Research. The views expressed are those of the authors and do not reflect the official policy or position of the funding agency or the U.S. Government. The research was in part carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton Univerisity Press, 2010.

[2] R. M. Murray, "Recent research in cooperative control of multivehicle systems," *Transactions-American Society of Mechanical Engineerers Journal of Dynamic Systems Measurement and Control*, vol. 129, no. 5, p. 571, 2007.

[3] T. Keviczky, F. Borrelli, and G. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006.

[4] T. Liu and Z.-P. Jiang, "A nonlinear small-gain approach to distributed formation control of nonholonomic mobile robots," in *American Control Conference (ACC)*. IEEE, 2013.

[5] 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.

[6] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, and S. Tsugawa, "Overview of platooning systems," in *Proceedings of the 19th ITS World Congress*, 2012.

[7] P. Kavathekar and Y. Chen, "Vehicle platooning: A brief survey and categorization," in *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. ASME, 2011, pp. 829–845.

[8] A. Levedahl, F. Morales, and G. Mouzakitis, "Platooning dynamics and control on an intelligent vehicular transport system," *CSOIS, Utah State University*, pp. 1–7, 2010.

[9] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549 – 558, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109806000136

[10] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, dec. 2004, pp. 4286 – 4291 Vol.4.

[11] T. M. Howard, "Adaptive model-predictive motion planning for navigation in complex environments," Ph.D. dissertation, Carnegie Mellon University, 2009.

[12] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 34, pp. 189 – 208, 1971. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370271900105

[13] M. Fox and D. Long, "Pddl2.1: An extension to pddl for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, p. 2003, 2003.

[14] Y. Q. Chen and Z. Wang, "Formation control: a review and a new consideration," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, pp. 3181–3186.

[15] M. Egerstedt and C. Martin, *Control theoretic splines: optimal control, statistics, and path planning*. Princeton University Press, 2009.