2002 Conference on Decision and Control

Real-Time Trajectory Generation for the Cooperative Path Planning of Multi-Vehicle Systems[1]

Feng-Li Lian[2] and Richard Murray
California Institute of Technology
Pasadena, CA 91125, USA
{fengli,murray}@caltech.edu

Abstract

This paper discusses a Cooperative Path Planning (CPP) design methodology for multi-vehicle systems and a Nonlinear Trajectory Generation (NTG) algorithm. Three scenarios of multi-vehicle tasking are proposed at the CPP framework. The NTG algorithm is, then, used to generate real-time trajectory for desired vehicle activities. Given system dynamics and constraints, the NTG algorithm first finds trajectory curves in a lower dimensional space and, then, parameterizes the curves by the B-spline basis. The coefficients of the B-splines are further solved by the sequential quadratic programming to satisfy the optimization objectives and constraints. The NTG algorithm has been implemented to generate real-time trajectories for a group of cooperative vehicles in the presence of changing missions and constraints.

1 Introduction

For large-scale autonomous vehicle systems, several distributed, hierarchical decompositions of controller algorithms have been proposed to overcome the problems in design complexity and computational limitation. The key feature of decomposing large-scale vehicle systems into a hierarchical architecture is that it translates a complicate controller design problem into several computationally tangible control sub-problems. A multi-layer planning, assessment, and control architecture of distributed semi-autonomous forces with collective objectives has been studied in the Mixed Initiative Control of Automa-teams (MICA) of DARPA. Conceptually, the MICA hierarchy includes Operations and Resources Supervisory for resource planning and human interaction, Team Composition and Tasking (TCT) for specifying group-level tasks, Team Dynamics and Tactics (TDT) for tasking team activities, Cooperative Path Planning (CPP) for generating feasible vehicle missions, and Vehicle Dynamics and Control. Planning and Control algorithms are accordingly designed to achieve functional goals specified at each layer [1].

Based on the above-mentioned hierarchies, a complex, difficult control problem can be properly decomposed into several sub-problems. Individual control algorithms can, then, be systematically designed to fulfill the sub-problem goals of one specified hierarchy, and the overall goal can be achieved by proper decomposition and construction techniques. For example, in a vehicle-routing case, one upper-layer controller might plan a grouping sequence of available vehicles and an assignment of feasible routes, and then generates an optimal activity plan for individual vehicles. Based on the planned activity received from the upper layer, the controller at lower layer is responsible for generating feasible trajectories in real time for each vehicle to follow. Therefore, multiple vehicles can utilize available resources and individually follow their own trajectories to achieve the overall system goal.

At the vehicle action planning layer, the CPP layer of MICA, one of the challenging problems is to plan and follow a trajectory in the presence of uncertainty and limited information. Limited information is due to the distributed nature of a multi-vehicle system and the range limitation of vehicle sensing and communication capabilities. To effectively control such systems, a two-degree-of-freedom design technique with a feedforward compensator and a feedback controller, as shown in Fig. 1, may be adopted. Based on the pre-defined goal, the feedforward compensator generates a nominal trajectory for the feedback controller to follow and produce proper actuation to the system input. Furthermore, the trajectory should be generated in real time and customized for the changes in mission, condition, and environment.

In this paper, we focus on the discussion of the design architecture and trajectory generation for cooperative vehicles at the CPP layer of MICA. The proposed CPP design architecture considers three scenarios of grouping and cooperation of unmanned air vehicles (UAV) flying from home base(s) to target(s) to deploy munitions and back. Based on desired missions and available information, the real-time trajectory is generated by the Nonlinear Trajectory Generation (NTG) algorithm that has been developed at Caltech [2, 3]. This paper consists of five sections, including the Introduction sec-

[1]Research supported in part by DARPA MICA program, Sharon Heise, Program Manager. [2] Corresponding author. Currently with Department of Electrical Engineering, National Taiwan University, Taipei, 106, Taiwan. E-mail: fengli@ccms.ntu.edu.tw

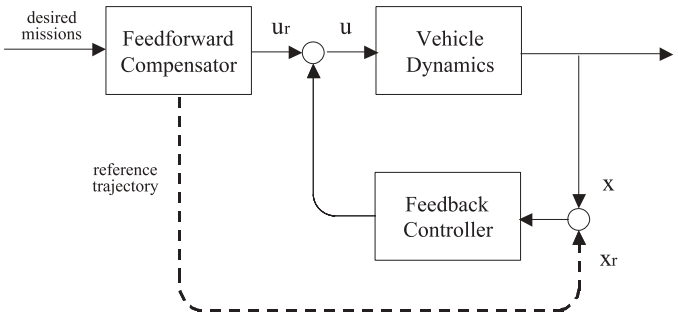
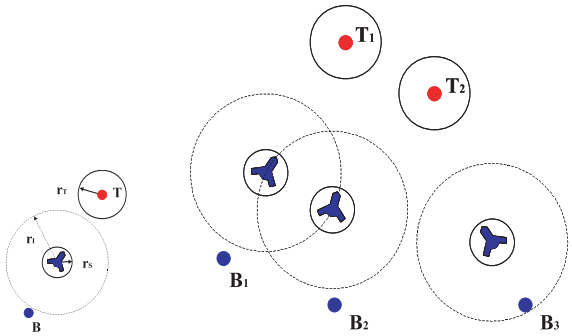


Figure 1: Two Degree of Freedom Design.



(a) (b)

Figure 2: The scenario of UAV tasking from home base(s) (B) to target(s) (T). r_S: safety radius, r_I: information radius, r_T: target detection radius.

tion. Section 2 describes the problem setup at the CPP layer. Section 3 outlines key components of the NTG algorithm. Section 4 presents the integration of the NTG algorithm into the CPP framework by providing an illustrative example of design procedures and trajectory generation. Summary and future directions are provided in the final section.

2 Problem formulation at CPP layer

In this section, we describe the problem formulation of the CPP layer of the MICA hierarchy. At the upper layer, the TCT controller plans and teams available resources such as vehicles and munitions to achieve specified group-level tasks. Taking the teaming results from the TCT controller as input, the TDT controller then generates a timing sequence of team activities. At the bottom, the CPP controller accepts the activity sequence from the TDT controller and generates feasible missions such as sets of waypoints and actions at these waypoints for individual vehicles. Operator commands and environmental uncertainty as well as the constraints of teaming and activity precedence, coordinated actions, and vehicle dynamics are also considered at the CPP layer. Hence, the controller design at CPP is to generate cooperative trajectories of one vehicle or a group of vehicles to support the planned activities as determined by the TDT controller. In the following, three scenarios of vehicle activities are discussed first, and the trajectory generation algorithm will be described in the next section.

Fig. 2 shows a scenario of UAV tasking from home base(s) (B) to target(s) (T). This scenario considers the case where multiple vehicles are commanded to accomplish designated activities. The target position and the designated action at the position is simply instructed by an upper-level command unit such as a TDT controller. After taking off from their home bases, UAVs need to compute real-time trajectories based on available information such as the target position, and the states of other UAVs and other adversarial entities and their threatening factors. As shown in Fig. 2(a), r_S denotes the safety region of the UAV and r_I represents the range of available sensing and communication information. For simplicity, we only consider the distance measures in two dimensional space. Having a relative distance larger than r_S, the UAV can safely fly without causing any damage. Hence, in order to succeed the desired missions, this constraint should be strongly imposed. On the other hand, r_I might be a combination of sensing capability to detect its neighboring environment, and communication capability of obtaining information from its neighboring vehicles. In general, $r_S < r_I$, otherwise, the UAV might collide with other units before it detects them or is informed by other units. Similarly, the target unit has a working radius of r_T that denotes a feasible detecting range if the target has a radar system or a threatening range if the target has a defensive capability.

In this case, three UAVs might be instructed by the same activity command, and need to fly together in a designated formation. Hence, the CPP controller at each individual UAV should generate a set of feasible, real-time trajectories which guarantee the group of vehicles to fly in the designated formation. A designated formation should keep the relative distance of any two UAVs be larger than r_S for collision avoidance and smaller than r_I for information sharing. Similar to the first case, r_T should be further considered when the group of UAVs are flying within the adversarial area.

In the next section, we describe the problem setup of the NTG algorithm. The integration of the NTG algorithm and the proposed CPP tasking will be presented in Section 4.

3 The NTG algorithm

In this section, we first outline the NTG algorithm and then describe its constructing techniques in detail. For a given system dynamics and a set of state and input constraints, and to minimize a pre-specified cost function, the NTG algorithm first makes use of the differential flatness property to find a new set of outputs in a lower dimensional space and then parameterizes the outputs by the B-spline basis representation. The coefficients of the B-splines are further solved by a sequential quadratic programming solver

to satisfy the optimization objectives and constraints. Finally, the trajectories for the vehicle controller to follow are represented by the B-spline curves with the obtained coefficients. In the following, we summarize the constructing techniques of the NTG algorithm presented in [2, 3]

Consider a nonlinear control system described as follows:

$$\dot{x} = f(x,u), \tag{1}$$

where $x \in \mathbb{R}^n$ are the states, $u \in \mathbb{R}^m$ are the inputs, and all vector fields and functions are assumed to be real-analytic. The states and inputs in system (1) is also assumed be to constrained by the following inequalities:

$$\begin{array}{rcccl} lb_0 & \leq & \psi_0(x(t_0),u(t_0)) & \leq & ub_0, \\ lb_f & \leq & \psi_f(x(t_f),u(t_f)) & \leq & ub_f, \\ lb_t & \leq & \psi_t(x(t),u(t)) & \leq & ub_t, \end{array} \tag{2}$$

where there are N_0 initial constraints, N_f final constraints, and N_t trajectory constraints, and lb's and ub's are their lower and upper bounds, respectively. In the UAV example, initial and final constraints might be imposed by the home base and target locations, and the trajectory constraints are induced from flight formation and adversarial environment. The problem is then to find a trajectory of system (1) that minimizes the following cost function:

$$J = \phi_0(x(t_0),u(t_0)) + \phi_f(x(t_f),u(t_f)) + \int_{t_0}^{t_f} L(x(t),u(t))dt, \tag{3}$$

where $\phi_0(\cdot,\cdot)$ and $\phi_f(\cdot,\cdot)$ are the costs associated with the initial and final locations, respectively, and $L(\cdot,\cdot)$ is the instant cost at time t.

The first step of the NTG algorithm is to determine a feasible set of outputs such that system (1) can be mapped into a lower dimensional output space. That is, it is desirable to find a set of outputs $z = \{z_1,...,z_q\}$ of the form:

$$z = G(x,u,u^{(1)},...,u^{(r)}), \tag{4}$$

such that (x,u) can be completely determined by z and its derivatives, i.e.,

$$(x,u) = H(z,z^{(1)},...,z^{(s)}), \tag{5}$$

where $u^{(i)}$ and $z^{(i)}$ denote the ith time derivative of u and z, respectively. A necessary condition for the existence of such outputs can be found in [4] and such systems are called differentially flat systems. If no flat outputs exist or one cannot find them, (x,u) can be still be completely determined by the following reduced-order form:

$$(x,u) = H_1(z,z^{(1)},...,z^{(s_1)}), 0 = H_2(z,z^{(1)},...,z^{(s_2)}). \tag{6}$$

In this case, an additional trajectory constraint needs to be included into the set of constraints (2).

Once a particular set of outputs are chosen, they are further parameterized in terms of the B-spline basis as follows [5]:

$$z_j(t) \quad = \quad \sum_{i=1}^{p_j} B_{i,k_j}(t) C_i^j \ \text{ for the knot sequence } \mathbf{t}_j,$$

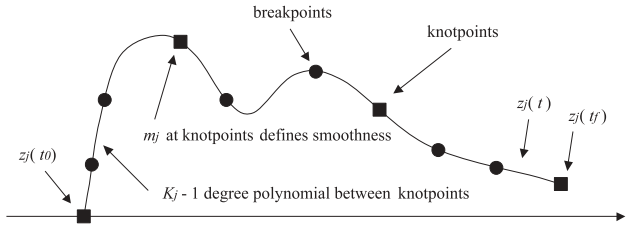


Figure 3: A B-Spline representation of $z_j(t)$.

where $B_{i,k_j}(t)$ are the ith B-spline basis function for the output $z_j(t)$ with polynomial order k_j, C_i^j are the coefficients of the B-splines. A B-spline representation of z_j with additional uniformly distributed breakpoints is pictured in Fig. 3. The total number of coefficients, p_j, can be computed by $p_j = l_j(k_j - m_j) + m_j$, where l_j denotes the number of knot intervals, m_j is the number of smoothness condition at the knot point.

After the outputs have been parameterized in terms of the B-spline curves, the cost function (3) and constraints (2) can also be re-formulated in terms of the coefficients of the chosen outputs; that is, $J(x,u) \rightarrow F(y)$ and $\{\psi_0(\cdot,\cdot),\psi_f(\cdot,\cdot),\psi_t(\cdot,\cdot)\} \rightarrow c(y)$, where $y = (C_1^1,...,C_{p_1}^1,C_1^2,...,C_{p_2}^2,...,C_1^q,...,C_{p_q}^q) \in \mathbb{R}^M$, $M = \sum_{i=1}^q p_i$. Note that $c(y)$ might also include the additional trajectory constraints as a result of not choosing a set of flat outputs. Hence, the problem can be formulated as the following nonlinear programming form:

$$\min_{y \in \mathbb{R}^M} F(y) \quad \text{subject to} \quad lb \leq c(y) \leq ub.$$

In NTG, the coefficients, i.e., y, of the B-spline curves are further solved by a sequential quadratic programming package, called NPSOL [6], to satisfy the optimization objective $F(y)$ and the constraints on $c(y)$. Finally, the state and input trajectories can be described in terms of these coefficients, and are fed into the feedback controller.

4 An illustrative example of flight formation

In this section, we use the scenario of flight formation of different UAVs to describe the integration of NTG algorithm into the MICA-CPP framework. As shown in Fig. 2, three UAVs are tasking from their home base B to target T. For the ease of presenting the design procedure, in this example, we consider a simplified 2-D UAV dynamics described as follows:

$$\dot{x}^i = u_x^i, \text{ and } \dot{y}^i = u_y^i, i = 1,2,3, \tag{7}$$

where x^i and y^i are the coordinate of UAV$_i$, and u_x^i and u_y^i are its corresponding inputs.

Furthermore, trajectory and input constraints are expressed

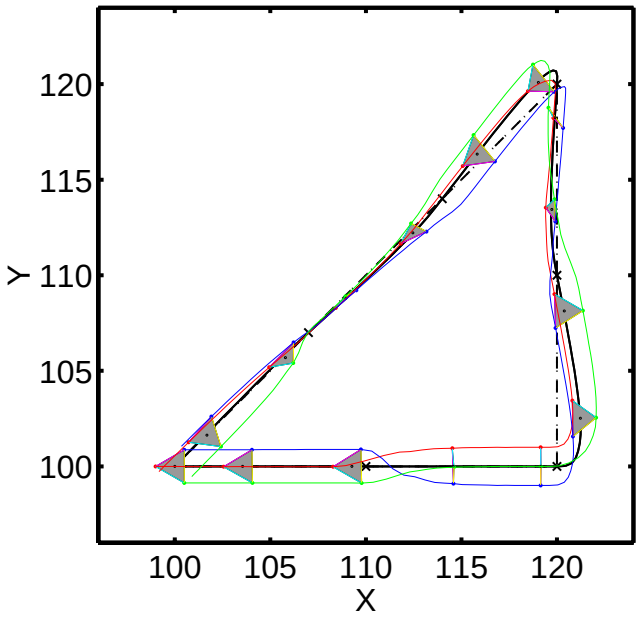


Figure 4: Flight formation of three UAVs.

as follows:

$$\begin{array}{ccccc} r_S & \leq & \sqrt{(x^i-x^j)^2+(y^i-y^j)^2} & \leq & r_I, \\ u^i_{lb:x,y} & \leq & u^i_x, u^i_y & \leq & u^i_{ub:x,y}, \end{array} \tag{8}$$

where $i, j = 1,2,3, i \neq j$, and the first inequality is for collision avoidance and obtaining information from its neighboring UAVs. The goal is assumed to task UAVs to the target by using minimal fuel and close formation. Hence, one choice of the cost function is as follows:

$$\begin{aligned} L(x,u) &= \sum_{i\neq j} \alpha_p^{ij} \left[\sqrt{(x^i-x^j)^2+(y^i-y^j)^2} - r_{12} \right]^2 \\ &+ \sum_{i=1}^{3} \alpha_p^{iR} \left[\sqrt{(x^i-x_R^i)^2+(y^i-y_R^i)^2} - r_{iR} \right]^2 \\ &+ \sum_{i=1}^{3} \alpha_u^i (u_x^i+u_y^i)^2, \end{aligned} \tag{9}$$

where α's are weighting factors, (x_R^i, y_R^i) is the reference trajectory specified by the upper-layer activity controller.

For this system, it is easy to find one set of flat outputs, $z_k, k = 1,...,6$ such that $x^i = z_{2\times i-1}$, $y^i = z_{2\times i}$ and $u_x^i = \dot{z}_{2\times i-1}$, $u_y^i = \dot{z}_{2\times i}$. For each output z_k, we let 'the number of intervals of knot points', 'the degree of smoothness at each knot point', and 'the polynomial degree' be 4, 3, 6, respectively. Hence, the number of coefficients of each output is 15 $(= 4(6-3)+3)$, that is, $z_k(t) = \sum_{i=1}^{15} B_{i,6}(t) C_i^k$ and $y = (C_1^1,...,C_{15}^1,...,C_{15}^6)$ in the nonlinear programming formulation.

One simulation study of different flight formations of three UAVs in two-dimensional space is shown in Fig. 4. Their base point is at (100,100) and multiple target points are located at (110,100), (120,100), (120, 110), (120, 120), (114,114), and (107,107). Hence, there are seven segments in the planning horizon. This group of UAVs change their flight formation at every target point and the sequence of the seven flight formations are "$\triangleright$", "$|$", "$\triangleright$", "$\backslash$", "$\triangleright$", "$/$", "$\triangleright$". That is, three UAVs first fly from (100,100) to (110,100) by using the "$\triangleright$" formation and change to the "$|$" formation at (110,100), and so on. In each segment, the simulation time is 5 seconds and 21 breakpoints are used. Different flight formations are coded by specifying different cost functions, but the constraint set remains the same. Collision avoidance during one formation or the changing of two formations is coded within the constraint set. In the beginning of each segment, the NTG algorithm is used to solve the optimal values of the coefficients of the flat outputs. The flying trajectory of each UAV is then constructed by the coefficients solved and their associated B-spline basis. Note that the exchanging of the UAV states needed in the NTG algorithm is performed only once when the flight formation changes.

5 Summary and future work

In this paper, we described the hierarchical design of large-scale multi-vehicle autonomous systems and discussed the scenario of vehicle tasking at the CPP layer of MICA. Based on a pre-designed vehicle activity, the trajectory for each vehicle to follow is then generated by the NTG algorithm. The constructing techniques of NTG was discussed in detail, and the integration of NTG into the MICA-CPP framework was also presented by an illustrative example of flight formation. In addition to the spatial constraints presented in this paper, our future work will focus on modifying the NTG algorithm and formulation to further incorporate temporal constraints such as activity coordination.

References

[1] Mixed Initiative Control of Automa-teams program of DARPA at http://dtsn.darpa.mil/ixo/mica.asp

[2] M. B. Milam, K. Mushambi, and R. M. Murray. A new computational approach to real-time trajectory generation for constrained mechanical systems. *2000 Conference on Decision and Control*, Dec. 2000.

[3] N. Petit, M. B. Milam, and R. M. Murray. Inversion based constrained trajectory optimization. *2001 IFAC symposium on Nonlinear Control Systems Design*, 2001.

[4] M. Fliess, J. Levine, P. Martin, and P. Rouchon. Flatness and defect of non-linear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327-1360, 1995.

[5] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[6] P. Gill, W. Murray, M. Saunders, and M. Wright. *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming.* System Optimization Laboratory, Stanford University, California, USA.