Cooperative Path Planning of Dynamical Multi-Agent Systems Using Differential Flatness Approach

Feng-Li Lian

Abstract: This paper discusses a design methodology of cooperative path planning for dynamical multi-agent systems with spatial and temporal constraints. The cooperative behavior of the multi-agent systems is specified in terms of the objective function in an optimization formulation. The path of achieving cooperative tasks is then generated by the optimization formulation constructed based on a differential flatness approach. Three scenarios of multi-agent tasking are proposed at the cooperative task planning framework. Given agent dynamics, both spatial and temporal constraints are considered in the path planning. The path planning algorithm first finds trajectory curves in a lower-dimensional space and then parameterizes the curves by a set of B-spline representations. The coefficients of the B-spline curves are further solved by a sequential quadratic programming solver to achieve the optimization objective and satisfy these constraints. Finally, several illustrative examples of cooperative path/task planning are presented.

Keywords: Cooperative path planning, differential flatness, multi-agent system, optimal trajectory generation.

1. INTRODUCTION

In recent years, dynamical multi-agent systems have been an active research area with advanced enabling technology for applications including tasks such as exploration in unknown area [1,2], military surveillance and reconnaissance [3], search and rescue [4,5], automated highway systems [6], formation control [7,8]. For this type of large-scale autonomous multi-agent 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 dynamical systems into a hierarchical architecture is that it translates a complicated controller design problem into several computationally tangible control sub-problems.

Research on Advanced Highway Systems (AHS), for example, proposes a hierarchical control architecture of five layers which decomposes a complex problem into several manageable units [6]. The five layers and their key functionalities are: (1) the network layer for deciding routes, (2) the link layer for assigning paths and target speeds, (3) the planning layer for managing maneuvers, (4) the regulation layer for completing tasks, and (5) the physical layer for controlling a vehicle itself. Vehicle control engineers can easily and systematically specify design requirements and goals, and design different controller algorithms for each individual layer. Similarly, 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) program of DARPA [9]. Conceptually, the MICA hierarchy includes operations and resources supervisory (ORS) 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 (VDC). Planning and control algorithms are accordingly designed to achieve functional goals specified at each layer. The layer decomposition of both AHS and MICA is briefly summarized in Fig. 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 agent-routing case, one

Manuscript received January 15, 2007; revised May 22, 2007 and December 7, 2007; accepted February 14, 2008. Recommended by Editorial Board member Jang Myung Lee under the direction of Editor Jae-Bok Song. This work was supported in part by the Mixed Initiative Control of Automa-teams program of DARPA, and the National Science Council, Taiwan, ROC, under the grants: NSC 95-2221-E-002-303-MY3, NSC 96-2218-E-002-030, and DOIT/TDPA: 95-EC-17-A-04-S1 -054.

Feng-Li Lian is with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: fengli@ntu.edu.tw).

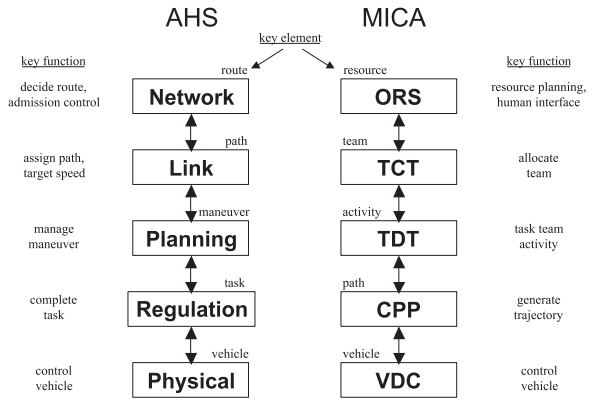


Fig. 1. The AHS and MICA hierarchies and their key elements and functions.

upper-layer controller might plan a grouping sequence of available agents and an assignment of feasible routes, and then generate an optimal activity for individual agent. 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 agent to follow. Therefore, multiple agents can utilize available resources and individually follow their own trajectories to achieve the overall system goal.

At the path planning layer, i.e., the regulation layer of AHS and the CPP layer of MICA, one of the challenging problems is to plan an optimal path for each agent within a team of dynamical multiagents. The path planning algorithms should deal with the dynamics of each agent as well as the motion interaction within the team of agents. To effectively control such systems, a two-degree-of-freedom design technique with a feedforward compensator and a feedback controller, as shown in Fig. 2, may be adopted. Based on the pre-defined goal, the feedforward compensator generates a set of nominal trajectories for the feedback controller of each agent to follow. Furthermore, the path should be generated in real time and customized for the changes in mission, condition, and environment. The class of problems can be viewed as a formation control problem, that is, the problem of controlling the relative pose of each agent in the team, while allowing the team to move as a whole [10].

Various approaches have been proposed for the formation control of a multi-agent system that can be classified into three categories: behavior-based, leader-following, and structured formations [11]. Behavior-based approaches first specify some simple motion primitives for each individual agent. Then, by implementing these motion primitives in terms of a reaction sequence, more complicated motions are produced through the dynamical interaction of multiple agents [12-14]. In leader-following approaches, one single agent or several multiple

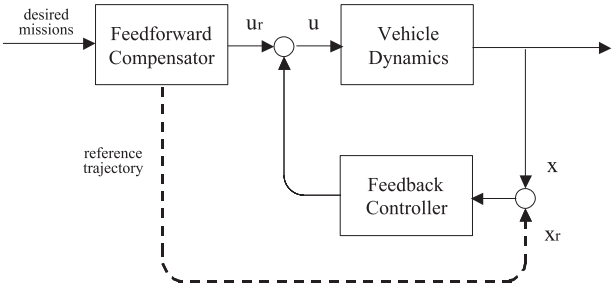


Fig. 2. Two degrees of freedom design.

agents play the role as the leader(s) and the other agents act as the followers of the designated leader(s). The key issues of the approach are the overall formation pattern generated by this set of local controllers and the robustness of control algorithm for maintaining the inter-agent position and velocity [10,15,16]. In a structured formation, the entire team of agents is treated as a single structure, e.g., virtual structure [17], and formation queue [18]. Desired formation is related to the designated structure which induces every individual path for these agents within the team to follow.

In the leader-following and structured formation approaches, planning an optimal path for each agent requires solving the dynamics of the whole set of agents under the formation constrains. On the other hand, the behavior-based approaches are difficultly analyzed mathematically. Therefore, the convergence of team formation cannot be guaranteed in advance [14]. Hence, in this paper, we introduce an optimal cooperative path planning algorithm based on differential flatness approach. Differential flatness is an intrinsic property of nonlinear control systems that can be used to transform the original complex dynamical equations into a set of algebraic equations. Moreover, the formation specification for achieving cooperative tasks is coded within the objective function. Therefore, standard optimization methodologies such as sequential quadratic programming can be used to solve the path planning problem. The advantages of the planning algorithm are two-fold. First, the computational complexity is reduced from solving numerical differential equations to dealing with a set of algebraic equations. Second, the cooperation, e.g., the designed formation pattern among dynamical multi-agents, is specified in terms of the objective function. That is, without modifying the dynamics of each individual agent, different cooperative paths can be easily generated.

In this paper, we focus on the discussion of the design architecture and plan planning for cooperative agents. The proposed design architecture considers three scenarios of grouping and cooperation of multiple agents. Based on desired missions and available information, the real-time path is generated by the planning algorithm. Given system dynamics

and state and input constraints, the algorithm first finds trajectory curves in a lower-dimensional space and, then, parameterizes the curves by the B-spline representation. The coefficients of the B-spline curves are further solved by a sequential quadratic programming solver to meet the optimization objectives and constraints. Finally, using the representation of these B-spline curves, the state and input trajectories are obtained to accomplish the designated activity. In order to incorporate the timing requirements in task planning, the actual timing variable is then redefined to become a new state variable and can be arbitrarily designed to fulfill any required temporal constraint. The actual running time will then be recovered from the solution of the optimization approach adopted.

This paper consists of six sections, including the Introduction section. Section 2 describes the problem formulation at the cooperative path planning framework. Section 3 discusses related mathematical background and Section 4 outlines key components of the planning algorithm. Section 5 presents the integration of temporal constraints into the algorithm. Section 6 provides illustrative examples of cooperative path/task planning in a three-agent system. Summary and future directions are provided in Section 7.

2. PROBLEM FOMULATION AT CPP LAYER

In this section, we describe the problem formulation of the cooperative path planning (CPP). At the upper layer of the MICA hierarchy as shown in Fig. 1, 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 agents. Operator commands and environmental uncertainty as well as the constraints of teaming and activity precedence, coordinated actions, and agent dynamics could also be considered at the CPP tasking. Hence, the controller design at CPP is to generate cooperative path of one agent or a group of agents to support the desired activities as determined by the TDT controller. In the following, three scenarios of agent activities are discussed first, and the path planning algorithm will be described in the next section.

Fig. 3 shows three scenarios of agents tasking from home bases (B) to targets (T). In Fig. 3(a), one single agent is tasking from the home base position to the target position. The target position and the designated action at the position is simply instructed by an upper-

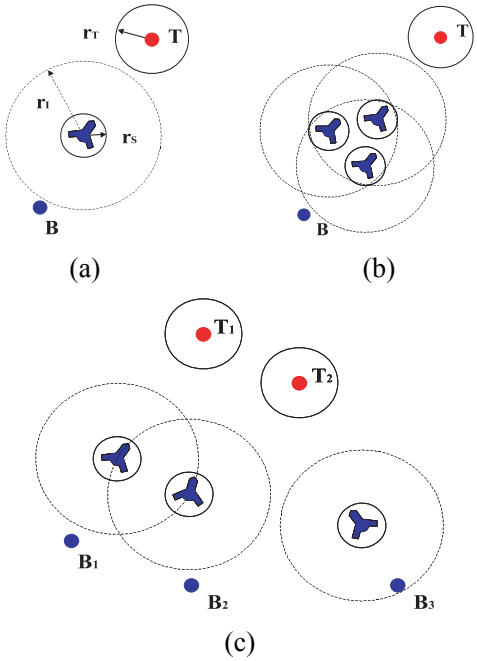


Fig. 3. Three scenarios of agents tasking from home bases (B) to targets (T). r_S : safety radius, r_I : information radius, r_T : target detection radius.

level command unit such as a TDT controller. After taking off from the home base, the agent needs to compute real-time paths based on available information such as the target position, the positions of other adversarial entities and their threatening factors, and its own state and input constraints. As shown in Fig. 3(a), r_S denotes the safety region of the agent and r_I represents the range of available sensing and communication information. For simplicity, the distance measures are in the two-dimensional space only. Having a relative distance larger than r_S, the agent can safely move 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 agents. In general, $r_S < r_I$, otherwise, the agent might collide with other agents before it detects them or is informed by others. 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.

The second case considers a scenario where multiple agents are commanded to accomplish a designated activity. For example, Fig. 3(b) shows that three agents are tasking from one home base to one

target location. In this case, three agents might be instructed by the same activity command, and need to move together in a designated formation. Hence, the CPP controller at each individual agent should generate a set of feasible, real-time paths which guarantee the group of agents to move in the designated formation. A designated formation should keep the relative distance of any two agents 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 agents are moving within the adversarial area.

The third case considers a more general scenario where multiple agents from different home bases are commanded to either one common target or multiple targets. At some location, these agents are commanded to move together and have a certain level of formation interaction. Conceptually, this scenario can be viewed as a combination of the first two cases. That is, when one agent just leaves its home base, its CPP controller works like that in the first case, and, when these agents are formed together, their CPP controllers work like those in the second case. However, more methodologies should be further developed in, for example, the merging and splitting of multiple agents.

In the next section, we first discuss related mathematical background of differential flatness for transforming the system model from differential equations into algebraic equations. The setup of the CPP algorithm, the integration of the CPP algorithm and the CPP tasking with temporal constraints will be presented in Sections 4 and 5.

3. MATHEMATICAL BACKGROUND

In this section, related mathematical background of the cooperative path planning algorithm is discussed. The key property and formulation of differentially flat systems are first outlined. A new set of functions, called flat outputs, can then be defined and used to describe the dynamical behavior of the system. Hence, a set of differential equations can be transformed into a set of algebraic equations. Similarly, the set of constraint equations for the dynamical system can also be represented by the same set of flat outputs. The set of algebraic equations along with the constraint equations can be easily solved by any standard nonlinear optimization solver.

Via a special feedback, a differentially flat system can be said to be equivalent to a linear system. Many realistic examples such as the crane, and the car with n trailers, are flat by properly choosing the state variables. Based on the discussion in [22], the mathematical properties of a differentially flat system are summarized as follows.

Assume that the dynamical model of one agent can be described by the following nonlinear control system:

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

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, and $f(0,0) = 0$ and

$$\text{rank } \frac{\partial f}{\partial u}(0,0) = m. \tag{2}$$

The states and inputs in system (1) are also assumed be to constrained within some state and input subspaces, U and X, that is, $u \in U$, $x \in X$. Hence, if the system is dynamically feedback linearizable, then the following regular dynamic compensator (a) and the diffeomorphism (b) can be found.

$$\begin{aligned} &\text{(a) } \dot{\eta} = a(x,\eta,v) \\ &\qquad u = b(x,\eta,v), \quad \eta \in \mathbb{R}^q, v \in \mathbb{R}^m, \end{aligned} \tag{3}$$

where $a(0,0,0)$, $b(0,0,0)$; and

$$\text{(b) } \xi = \Xi(x,\eta), \quad \xi \in \mathbb{R}^{n+q}. \tag{4}$$

Hence, the closed-loop system can be transformed into a linear controllable system, represented by the new state variables ξ, that is, $\dot{\xi} = F\xi + Gv$, where F, G are constant matrices of compatible dimension. Moreover, the linear system can be further transformed into the Brunovsky canonical form

$$\begin{aligned} z_1^{(n_1)} &= v_1, \\ z_2^{(n_2)} &= v_2, \\ &\vdots \\ z_m^{(n_m)} &= v_m, \end{aligned} \tag{5}$$

where $n_1, n_2, \ldots, n_m$ are the controllability indices and $Z = (z_1, \cdots, z_1^{(n_1-1)}, \cdots, z_m, \cdots, z_m^{(n_m-1)})$ is another set of basis vectors spanned by the new state variables ξ. Hence, one invertible $(n+q)\times(n+q)$ matrix T can be obtained, such that $Z = T\xi$, or $Z = T\,\Xi(x,\eta)$. Therefore, by the diffeomorphism property of Ξ, x and ξ can be regarded as functions of the new basis Z. Furthermore, the input u can also be regarded as a function of Z and v. Finally, the original state x and the input u can be expressed as functions of Z as follows.

$$\begin{aligned} x &= A(z, \dot{z}, \cdots, z^{\alpha}), \\ u &= B(z, \dot{z}, \cdots, z^{\beta}), \end{aligned} \tag{6}$$

where $z = (z_1, z_2, \cdots, z_m)$ and α, β are some

constants.

In summary, a dynamical system is said to be differentially flat if it is linearizable via the special feedback where the new set of basis z is regarded as a fictitious output and called a flat output. By utilizing the differentially flatness property, the state and input variables can be directly described as functions of the flat output and a finite number of its derivatives. Hence, the problem of generating proper trajectory for the state and input can be transformed into the problem of the trajectory generation of the flat output z. Mathematically speaking, by assigning the new output variables, the problem of solving a set of differential equations can be transformed into that of solving a set of algebraic equations.

For the path planning of cooperative multi-agents, the dynamics of the agent along with the constraints on the state and input variables, and their cooperation can be formulated as an optimization problem. The cooperative functionality is mathematically formulated as an integrated objective function. Assume that the dynamical model of one agent can be described by the differential equation (1). The state and input are also assumed be to constrained by the following inequalities:

$$\begin{aligned} &L_0 \le c_0(x(t_0),u(t_0)) \le U_0, \\ &L_f \le c_f(x(t_f),u(t_f)) \le U_f, \\ &L_t \le c_t(x(t),u(t)) \le U_t, \ \ t_0 \le t \le t_f, \end{aligned} \tag{7}$$

where $c_*(\cdot,\cdot)$ is a function of x,u; t_0,t_f are the initial and final times, respectively; and L_*'s,U_*'s represent the lower and upper bounds, respectively, of the constraints. Assume that there are N_0 initial constraints, N_f final constraints, and N_t path constraints. The initial and final constraints might be imposed by the home base and target locations, and the path constraints are induced from the agent formation and adversarial environment. The problem is then to find a set of paths for system (1) that minimizes the following objective function:

$$\begin{aligned} J = &o_0(x(t_0),u(t_0)) + o_f(x(t_f),u(t_f)) \\ &+ \int_{t_0}^{t_f} o_t(x(t),u(t))dt, \end{aligned} \tag{8}$$

where $o_0(\cdot,\cdot)$ and $o_f(\cdot,\cdot)$ are the objective functions associated with the initial and final locations, respectively, and $o_t(\cdot,\cdot)$ is the instant objective function at time t. Furthermore, typical components of $o_t(\cdot,\cdot)$ are described as follows:

$$o_t(x,u) = o_r(x,u) + o_f(x,u) + o_a(x,u), \tag{9}$$

where $o_r(\cdot,\cdot)$ denotes the cost of tracking one reference path, $o_f(\cdot,\cdot)$ denotes the cost of maintaining one designated formation pattern, and $o_a(\cdot,\cdot)$ denotes the cost associated with the agent itself such as the fuel used [21]. Note that the cooperation of multiple agents is enforced by including different formation functions in $o_f(\cdot,\cdot)$.

By utilizing the differential flatness property, the coupled dynamics of the multiple agents are transformed into a set of algebraic equations in terms of a set of new output variables in a lower-dimensional space. Similarly, the set of constraints can also be represented by the set of new output variables. Finally, the set of algebraic equations along with the constraints can be solved by any standard nonlinear programming solver. Hence, the optimal solution for the new output variables can be found and the optimal curves of the original state and input variables can then be obtained by (6).

4. THE COOPERATIVE PATH PLANNING ALGORITHM

In this section, we first outline the CPP algorithm and then describe its related constructing techniques in detail. For a given system dynamics and a set of state and input constraints, and to minimize a pre-specified objective function, the CPP 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 set of the B-spline basis functions. The coefficients of the B-spline curves are further solved by a sequential quadratic programming solver to meet the optimization objectives and constraints. Finally, the path for the agent controller to follow is represented by the B-spline curves with the obtained coefficients [19,20].

The first step of the 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 flat outputs $z = \{z_1,\cdots,z_q\}, q \le n,$ of the form:

$$z = g(x,u,u^{(1)},\cdots,u^{(r)}), \tag{10}$$

that is, z is a function of $x,u,u^{(1)},\cdots,u^{(r)}$, such that (x,u) can be completely determined by (10), i.e.,

$$(x,u) = h(z,z^{(1)},\cdots,z^{(s)}), \tag{11}$$

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 [22] 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:

$$(x,u) = h_1(z, z^{(1)}, \cdots, z^{(s_1)}) \quad \text{and} \tag{12}$$

$$0 = h_2(z, z^{(1)}, \cdots, z^{(s_2)}). \tag{13}$$

In this case, an additional path constraint, i.e., (13), should be included into the set of constraints (7).

Once a particular set of outputs are chosen, they are further parameterized in terms of the B-spline curves as follows [23]:

$$z_1(t) = \sum_{i=1}^{p_1} b_{i,k_1}(t) C_i^1 \quad \text{for the knotpoint sequence } \mathrm{t}_1,$$

$$z_2(t) = \sum_{i=1}^{p_2} b_{i,k_2}(t) C_i^2 \quad \text{for the knotpoint sequence } \mathrm{t}_2,$$

$$\vdots$$

$$z_q(t) = \sum_{i=1}^{p_q} b_{i,k_q}(t) C_i^q \quad \text{for the knotpoint sequence } \mathrm{t}_q,$$

where $b_{i,k_j}(t)$ are the i-th B-spline basis functions for the output z_j with order k_j, C_i^j are the coefficients of the B-spline. $p_j = l_j \cdot (k_j - m_j) + m_j$ where l_j is the number of knotpoint intervals, and m_j is the number of smoothness condition at the knotpoint. A B-spline representation of z_j with additional uniformly distributed breakpoints is pictured in Fig. 4.

After the outputs have been parameterized in terms of the B-spline curves, the objective function (8) and constraints (7) can also be re-formulated in terms of the coefficients of the chosen outputs; that is, $J(x,u) \to \bar{J}(y)$ and $\{c_o(\cdot,\cdot), c_f(\cdot,\cdot), c_t(\cdot,\cdot)\} \to \bar{c}(y)$ where $y = (C_1^1, \cdots, C_{p_1}^1, C_1^2, \cdots, C_{p_2}^2, \cdots, C_1^q, \cdots, C_{p_q}^q) \in \mathbb{R}^M$, $M = \sum_{i=1}^{q} p_i$. Note that $\bar{c}(y)$ might also include the additional path constraints as a result of not choosing a set of flat outputs. Hence, the problem can be formulated as the following nonlinear programming form:

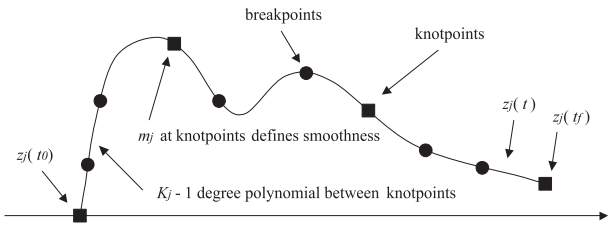


Fig. 4. A B-Spline representation of z_j.

System Dynamics:

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

State and Input Constraints:

$$\begin{array}{rcccl} L_0 & \le & c_0(x(t_0),u(t_0)) & \le & U_0 \\ L_f & \le & c_f(x(t_f),u(t_f)) & \le & U_f \\ L_t & \le & c_t(x(t),u(t)) & \le & U_t \end{array}$$

Cost Function:

$$J = o_0(x(t_0),u(t_0)) + o_f(x(t_f),u(t_f)) + \int_{t_0}^{t_f} o_t(x(t),u(t))dt$$

New Outputs:

$$z = g(x,u,u^{(1)},...,u^{(r)}), \quad (x,u) = h(z,z^{(1)},...,z^{(s)})$$

Using B-spline representation:

$$z_1(t) = \sum_{i=1}^{p_1} b_{i,k_1}(t) C_i^1, ..., z_q(t) = \sum_{i=1}^{p_q} b_{i,k_q}(t) C_i^q$$

New Cost Function:

$$\min_{y \in \mathbf{R}^M} \bar{J}(y) \quad \text{subject to} \quad L \le \bar{c}(y) \le U$$

$$y = (C_1^1, ..., C_{p_1}^1, C_1^2, ..., C_{p_2}^2, ..., C_1^q, ..., C_{p_q}^q)$$

Fig. 5. Key formulation of the cooperative path planning.

$$\min_{y \in \mathbb{R}^M} \bar{J}(y) \quad \text{subject to} \quad L \le \bar{c}(y) \le U. \tag{14}$$

The coefficients y of the B-spline curves are further solved by a sequential quadratic programming package, called NPSOL [24], to satisfy the optimization objective $\bar{J}(y)$ and the constraints on $\bar{c}(y)$. Finally, the state and input trajectories can be described in terms of these coefficients, and are fed into the feedback controller.

The formulation is implemented in the formation control of one multi-agent team where the formation is specified within the objective function along with other state and input constraints. That is, any spatial constraints on the team tasking can be easily coded into the constraint set, (7). Key formulation of the cooperative path planning algorithm is summarized in Fig. 5. Applications of the differential flat approach to mechanical examples such as 2-D crane, car with n-trailers, the Kapitsa pendulum, and the inverted double pendulum can be found in [22]. Also, the examples of a planar ducted fan and unicycle are discussed in [19] and [25], respectively.

5. INTEGRATING TEMPORAL CONSTRATIONS

In order to further include any temporal constraint associated to agent activities, the original formulation should be modified to augment one additional time variable into each agent dynamics [19]. We first define a slack state variable T and let $T = t/\tau$, where t and τ are "old" and "new" time variables,

respectively. That is, T denotes the ratio between the true time variable and the new pseudo time variable. By doing so and letting the time interval of τ be [0,1], the algorithm can generate a path for the pseudo time from $\tau = 0$ to $\tau = 1$ and numerically solve the value of T simultaneously. Hence, including the slack time variable T, the agent dynamics is augmented as follows:

$$x' = f(x,u,T), \tag{15}$$

$$T' = 0, \tag{16}$$

where $(\cdot)' = d(\cdot)/d\tau$, that is, the new dynamics is formulated in terms of the new pseudo time t and will be numerically solved for $\tau \in [0,1]$. Furthermore, the set of state and input constraints and additional temporal constraints can be expressed by the following set of inequalities:

$$\begin{aligned} &L_0 \le \hat{c}_0(x(0),u(0),T) \le U_0, \\ &L_f \le \hat{c}_f(x(1),u(1),T) \le U_f, \\ &L_\tau \le \hat{c}_\tau(x,u,T) \le U_\tau, \\ &L_T \le \hat{c}_T(T) \le U_T. \end{aligned} \tag{17}$$

The introduction of new time variables τ and T could also change linear constraints into nonlinear constraints. For example, consider the following the constraints on initial velocity and acceleration:

$$\begin{aligned} &L_v \le \dot{x} \le U_v, \\ &L_a \le \ddot{x} \le U_a. \end{aligned}$$

Using the definitions of $T = t/\tau$ and $(\cdot)' = d(\cdot)/d\tau$, the above two inequalities become the following nonlinear constraints:

$$\begin{aligned} &L_v \le x'/T \le U_v, \\ &L_a \le x''/T \le U_a, \end{aligned}$$

because both x and T are variables. Also, the objective function of the augmented systems can be modified as follows:

$$\begin{aligned} J = {} & \hat{o}_0(x(0),u(0),T) + \hat{o}_f(x(1),u(1),T) \\ & + \int_0^1 \hat{o}_\tau(x(\tau),u(\tau),T)d\tau. \end{aligned} \tag{18}$$

Note that the initial and final times (of τ) of the integration have been changed from (t_0, t_f) to (0,1), and the actual final time, t_f, is equivalent to $t_f = T$ since $T = t/\tau$ and $\tau = 1$. After this modification, we can construct temporal constraints as well as spatial constraints in the CPP algorithm directly. The cooperative path can then be generated based on

System Dynamics:

$$\begin{aligned} x' &= f(x,u,T) \\ T' &= 0 \\ \text{where } ()' &= d()/d\tau \text{ and } T = t/\tau \end{aligned}$$

State, Input, Temporal Constraints:

$$\begin{aligned} L_0 &\le \hat{c}_0(x(0),u(0),T) \le U_0 \\ L_f &\le \hat{c}_f(x(1),u(1),T) \le U_f \\ L_\tau &\le \hat{c}_\tau(x,u,T) \le U_\tau \\ L_T &\le \hat{c}_T(T) \le U_T \end{aligned}$$

Cost Function:

$$\begin{aligned} J &= \hat{o}_0(x(0),u(0),T) + \hat{o}_f(x(1),u(1),T) \\ &+ \int_0^1 \hat{o}_\tau(x(\tau),u(\tau),T)Td\tau \end{aligned}$$

Call CPP in Fig. 5

Fig. 6. Key formulation of the cooperative path planning with temporal constraints.

different pre-specified planning time horizons of each agent activity. Key formulation of the cooperative path planning algorithm with temporal constraints is summarized in Fig. 6.

In practical design situation of a multi-agent system, the number of agents could be large and the total dimension of agent dynamics could be big. Also, each agent might have multiple tasks that need to be coordinated with those of other agents. Hence, the computational complexity in the agent-task space could be in the order of $n \times N \times L$, where n is the dimension of agent dynamics, N is the number of agents, and L is the number of tasks of each agent required to perform. If the total number of agents involved in the task planning is too large, the algorithm might spend longer computational time to find an optimal solution. This drawback can be overcome by imposing planning time window for each agent-task, that is, adding one extra timing constraint in the fourth inequality of (17). Therefore, the task planning and path generation of each agent can be done separately. However, a high-level task planner is needed to generate the time window for each agent-task, and the temporal constraints in this case will be more conservative compared with the previous case. Two illustrative examples of the cooperative path planning of three agents are presented in the next section.

6. ILLUSTRATIVE EXAMPLES

6.1. Agent formation with spatial constraints

In this section, we use the formation scenario of three agents to describe the cooperative path planning task [27]. As shown in Fig. 3, three agents are tasking from their home base B to target T. For the ease of presenting the design procedure, a simplified 2-D model of agent dynamics is described as follows:

$$\dot{x}^i = u_x^i, \quad \text{and} \quad \dot{y}^i = u_y^i, \quad i = 1,2,3, \tag{19}$$

where x^i and y^i are the coordinates of the ith agent, and u_x^i and u_y^i are its corresponding inputs. Furthermore, path and input constraints are expressed as follows:

$$\begin{aligned} & r_S \le \sqrt{(x^i - x^j)^2 + (y^i - y^j)^2} \le r_I, \\ & u_{lb:x,y}^i \le u_x^i, u_y^i \le u_{ub:x,y}^i, \end{aligned} \tag{20}$$

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

$$\begin{aligned} o_t(x,u) = & \sum_{i \ne j} \alpha_p^{ij} \left[\sqrt{(x^i - x^j)^2 + (y^i - y^j)^2} - r^{ij} \right]^2 \\ & + \sum_{i=1}^{3} \alpha_p^{iR} \left[\sqrt{(x^i - x_R^i)^2 + (y^i - y_R^i)^2} - r^i \right]^2 \\ & + \sum_{i=1}^{3} \alpha_u^i (u_x^i + u_y^i)^2, \end{aligned} \tag{21}$$

where α's are weighting factors, (x_R^i, y_R^i) is the reference path specified by the upper-layer activity controller, and r^{ij}, r^i are desired ranges between agents as well as each agent and the path. Note that the first summation in (21) denotes the cost of maintaining one designated formation pattern, the second summation denotes the cost of tracking one reference path, and the last summation denotes the cost (or fuel) for controlling the agent. Also, the cooperation of maintaining a proper formation is enforced by the first set of cost functions. Therefore, if α_p^{ij} is larger, then the objective is to maintaining a predefined formation. On the other hand, if α_p^{iR} is larger, each agent has strong potential on tracking a desired trajectory. Finally, if the control input (i.e., fuel consumption) is more important, then α_u^i should be set as a larger value compared with the other two sets of parameters.

For this system, it is easy to find one set of flat outputs, $z_k, k = 1, \cdots, 6$, 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 knotpoints', 'the degree of smoothness at each knotpoint', 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 $y = (C_1^1, \cdots, C_{15}^1, \cdots, C_{15}^6)$ in the nonlinear programming formulation.

One simulation study of different formations of three agents in a two-dimensional space is shown in Fig. 7. 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 planning horizons. This group of agents change their formation at every target point and the sequence of the seven formations are "C", "|", "C", "\", "C", "/", "C". That is, three agents first fly from (100,100) to (110,100) by using the "C" formation and change to the "|" formation at (110,100), and so on. In each segment, the simulation time is set as 5 seconds because, during this time period, a good dynamical behavior of the agent team can be observed. Also, 21 breakpoints are used in each segment. In this case, selecting the number of points should depend on the smoothness of generated trajectory as well as

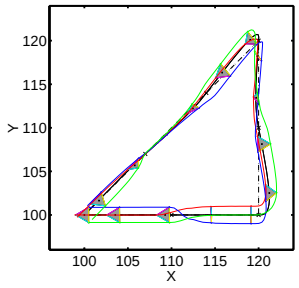


Fig. 7. Formation of three agents.

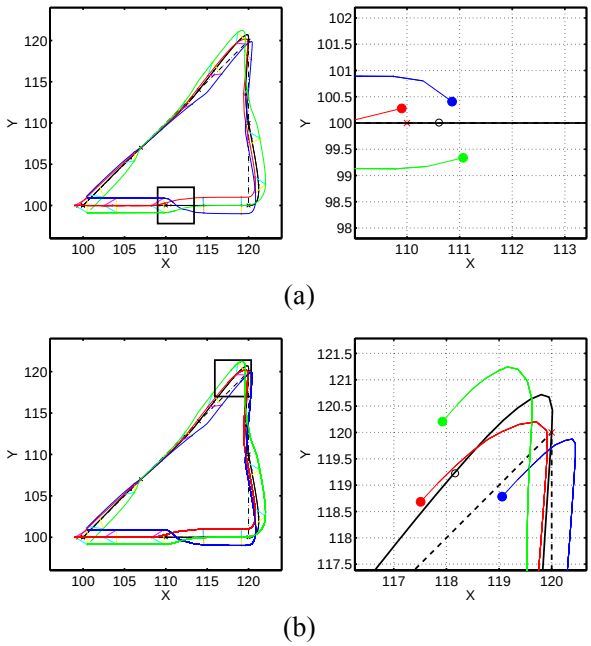


Fig. 8. Snapshots of simulation result of formation change.

computational cost. If a large number of breakpoints is selected, then the generated trajectory is smooth, but the computational cost increases. Different formations are coded by specifying different objective functions, that is, different settings for the first summation in (21), but the set of constraints remain the same. Also, the dynamical location of the formation is based on a pre-defined set of desired paths and enforced by the second summation in (21). Collision avoidance during one formation is coded within the constraint set. In the beginning of each segment, the algorithm solves the optimal values of the coefficients of the flat outputs. The path of each agent is then constructed by the coefficients solved and their associated B-spline basis. Fig. 8 shows two snapshots of the simulation result near (110,100) and (120,120) during the formation change. The switching of formation patterns is done by setting functions of different related distance between any pair of agents.

6.2. Path planning based on different dynamical models

In order to illustrate the effectiveness and convenience of using the proposed framework for generating cooperative paths, three different dynamical models: namely, direct, kinematic, and dynamic, are added in the section. For the ease of presenting the design procedure, three agents operated in a 2-D plane are considered. First, similar to that discussed in Section 6.1, the direct model of agent dynamics is described as follows:

$$\begin{aligned} \dot{x}^i &= u_x^i, \\ \dot{y}^i &= u_y^i, \quad i = 1,2,3, \end{aligned} \tag{22}$$

where x^i and y^i are the coordinates of the ith agent, and and u_x^i and u_y^i are its corresponding inputs. Second, the kinematic model of agent dynamics is described as follows:

$$\begin{aligned} \dot{x}^i &= v^i \cos(\theta^i), \\ \dot{y}^i &= v^i \sin(\theta^i), \\ \dot{\theta}^i &= w^i, \qquad i = 1,2,3, \end{aligned} \tag{23}$$

where x^i and y^i are the coordinates of the ith agent, θ^i is the orientation of the ith agent, and v^i and w^i are the translational and rotational velocity inputs, respectively. Third, the dynamical model of agent dynamics is described as follows:

$$\begin{aligned} m^i \ddot{x}^i + \eta^i \dot{x}^i &= (F_s^i + F_p^i)\cos(\theta^i), \\ m^i \ddot{y}^i + \eta^i \dot{y}^i &= (F_s^i + F_p^i)\sin(\theta^i), \\ J^i \ddot{\theta}^i + \phi^i \dot{\theta}^i &= (F_s^i - F_p^i) r^i, \qquad i = 1,2,3. \end{aligned} \tag{24}$$

This example is the dynamical model of a vehicle with two ducted fans for propulsion studied in [26]. Hence, F_s^i, F_p^i are two inputs and should be positive, and m^i, η^i, J^i, ϕ^i, and r^i are mass, viscous friction, rotational inertia, rotational friction, and moment arm of these fan inputs, respectively.

The goal is to generate paths for forming a close right triangular formation. Hence, the objective function is chosen as follows:

$$o_t(x,u) = \sum_{i \neq j} \left[\sqrt{(x^i - x^j)^2 + (y^i - y^j)^2} - r^{ij} \right]^2, \tag{25}$$

where r^{ij}'s are desired relative distance between agents. For the first direct model, it is easy to find one set of flat outputs, $z_k, k = 1, \cdots, 6$, such that $x^i = z_{2\times i-1}, y^i = z_{2\times i}$ and $u_x^i = \dot{z}_{2\times i-1}, u_y^i = \dot{z}_{2\times i}$. Hence, two flat outputs are used for each agent. For the second model, the following nonlinear constraint can be found based on the kinematic model.

$$\dot{x}^i \sin(\theta^i) = \dot{y}^i \cos(\theta^i), \qquad i = 1,2,3. \tag{26}$$

For the case, although there are three physical outputs at each agent, i.e., x^i, y^i, θ^i, only two flat outputs, $x^i = z_{2\times i-1}, y^i = z_{2\times i}$, are enough to represent the system variables. The third output, θ^i, can be computed from (26), that is,

$$\theta^i = \tan^{-1}(\dot{y}^i / \dot{x}^i), \qquad i = 1,2,3. \tag{27}$$

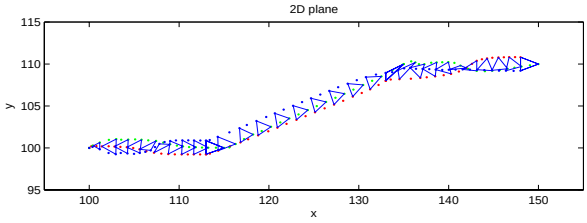
Similarly, for the third dynamic model, the following nonlinear constraint can be found.

$$(m^i \ddot{x}^i + \eta^i \dot{x}^i)\sin(\theta^i) = (m^i \ddot{y}^i + \eta^i \dot{y}^i)\cos(\theta^i). \tag{28}$$

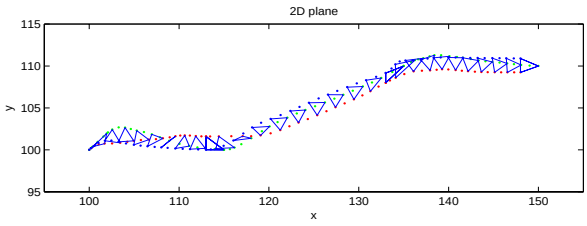
Therefore, although there are three physical outputs at each agent, i.e., x^i, y^i, θ^i, only two flat outputs, $x^i = z_{2\times i-1}, y^i = z_{2\times i}$, are enough to represent the system variables. The third output, θ^i, can be generated based on the calculation from (28). Also, in the case, the fan inputs for the agent should be positive. Hence, based on the dynamical model (24), the representation of these fan inputs is formulated as follows:

$$\begin{aligned} F_s^i &= \frac{1}{2}\left[(m^i \ddot{x}^i + \eta^i \dot{x}^i)\cos(\theta^i) + (m^i \ddot{y}^i + \eta^i \dot{y}^i)\sin(\theta^i) + \frac{1}{r^i}(J^i \ddot{\theta}^i + \phi^i \dot{\theta}^i) \right], \\ F_p^i &= \frac{1}{2}\left[(m^i \ddot{x}^i + \eta^i \dot{x}^i)\cos(\theta^i) + (m^i \ddot{y}^i + \eta^i \dot{y}^i)\sin(\theta^i) - \frac{1}{r^i}(J^i \ddot{\theta}^i + \phi^i \dot{\theta}^i) \right], \\ & \qquad i = 1,2,3. \end{aligned} \tag{29}$$

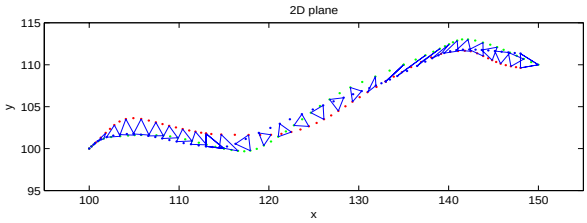
Therefore, $F_s^i \geq 0$ and $F_p^i \geq 0$ can be easily formulated as inequality constraints in terms of flat outputs.



(a) Direct model.



(b) Kinematic model.



(c) Dynamical model with input constraints.

Fig. 9. Formation of three agents based on different dynamics models.

The simulation results of these three cases are shown in Fig. 9(a)-(c). Three segments of desired paths are considered in all the cases. These desired paths are started from (100,100), going through (115, 100), and (135, 110), and stopped around (150, 110). In each segment, each path consists of 21 generated points, and the triangular formations of these agents shown in the figure are for every other points. As shown in Fig. 9(a) and (b), the paths generated by the direct and kinematic models have similar characteristics and fit into the desired paths closely. However, the paths generated by the dynamic model have higher-order maneuver due to the second-order dynamics as well as the positive input constraints.

6.3. Agent formation with temporal constraints

In this section, we use the scenario of activity coordination of different agents with temporal constraints, as shown in Fig. 10 [28], and detailed description of these activities is summarized in Table 1. Simply speaking, three agents are routed in a manner to achieve a set of coordinated activities (a_{ij}, denoting the jth activity of ith agent). For example, two scenarios are considered: The 'look' activity a_{12} of Agent 1 on Object b must happen after the 'strike' activity a_{21} of Agent 2, and there is a simultaneous 'strike' activity by Agents 1 and 3 on Object c. The two sets of coordinated activities can be formulated as the following temporal constraints:

$$T^{11} + T^{12} \geq T^{21}, \tag{30}$$

$$T^{11} + T^{12} + T^{13} = T^{31} + T^{32}, \tag{31}$$

where T^{ij} denotes the planning time horizon of the jth activity of the ith agent.

For the ease of presenting the design procedure, a simplified 2-D model of agent dynamics is described as follows:

$$\dot{x}^{ij} = u_x^{ij}, \text{ and } \dot{y}^{ij} = u_y^{ij}, \quad i = 1,2,3, \tag{32}$$

where x^{ij} and y^{ij} are the coordinates of the jth activity of the ith agent, and u_x^{ij} and u_y^{ij} are its corresponding inputs.

Additional state and input constraints can be further expressed as follows:

$$\begin{aligned} & r_S \leq \sqrt{(x^{ij} - x^{kj})^2 + (y^{ij} - y^{kj})^2} \leq r_I, \\ & u_{lb:x,y}^{ij} \leq u_x^{ij}, u_y^{ij} \leq u_{ub:x,y}^{ij}, \end{aligned} \tag{33}$$

where $i, k = 1,2,3, i \neq k$, and the first inequality is for collision avoidance and the range of obtaining information from its neighboring agents. The goal is assumed to task the three agents to the target by using minimal fuel and time. Hence, one choice of the objective function is as follows:

$$o_\tau(x,u) = \sum_{ij} \alpha_T^{ij} (T^{ij})^2 + \alpha_u^{ij} (u_x^{ij} + u_y^{ij})^2, \tag{34}$$

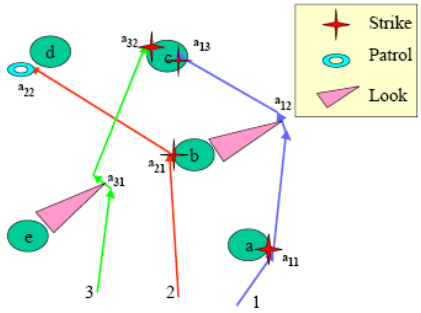


Fig. 10. Activity coordination of three agents.

Table 1. Planned activities of these three agents at five targets.

Target	home	a	b	c	d	e
Agent 1	v1 → ←	P,S,P → ←	L → L ←	P,S,P		
Agent 2	v2 → ←	→	P,S,P	→ ←	P	
Agent 3	v3 → ←			P,S,P ←	→ ← → ←	L L

P: Patrol, S: Strike in and out, L: Look

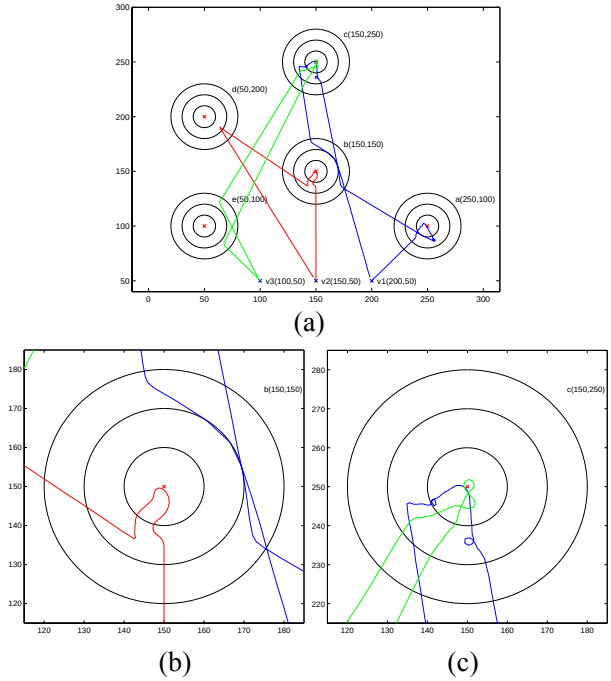


Fig. 11. Activity coordination of three agents with additional tasking around target points. (a) the whole scenario, (b) a close look around (150,150) when Agent 1 looks at Target b after Agent 2 strikes it, (c) a close look around (150,250) when Agent 1 and Agent 3 strike Target c simultaneously.

For this system, it is easy to find one set of flat outputs, z_k such that $(x^{ij}, y^{ij}, T^{ij}, u_x^{ij}, u_y^{ij}) = (z_k, \dot{z}_k)$, and to implement the parametrization $z_k(t) = \sum_{i=1}^{15} b_{i,6}(t) C_i^k$ and $y = (C_i^k)$ in the nonlinear programming formulation.

The simulation results of activity coordination of three agents in a two-dimensional space are shown in Fig. 11. The three agents are based at at (200,50), (150,50), and (100,50), respectively, and multiple target points are located at a(250,100), b(150,150), c(150, 250), d(50,200), and e(50,100). At each target point, three circles of different radii are depicted to schematically indicate different activities occurring at the target point. These three regions, started from the target points, are denoted as 'Strike', 'Patrol', and 'Look' areas. In Fig. 11, solid lines are the generated path of these three agents and small dots are the points generated by the algorithm. Particularly, Fig. 11(b) shows the scenario that Agent 1 looks at Target b after Agent 2 strikes it and Fig. 11(c) shows the scenario that Agent 1 and Agent 3 strike Object c simultaneously.

7. SUMMARY AND FUTURE WORK

In this paper, we described the hierarchical design of large-scale multi-agent systems and discussed the scenario of agent tasking at the cooperative path planning framework. Based on a pre-designed agent activity, the path for each agent to follow is then generated by the CPP algorithm. The constructing techniques of the algorithm were discussed in detail, and the integration of algorithm into the CPP framework was also presented by illustrative examples. In addition to the spatial constraints, the incorporation of temporal constraints such as activity coordination was discussed in this paper. The advantages of the planning algorithm are two-fold. First, the computational complexity is reduced from solving numerical differential equations to dealing with a set of algebraic equations. Second, the cooperation, e.g., the designated formation pattern among dynamical multi-agents, is specified in terms of the objective function. That is, without modifying the dynamics of each individual agent, different cooperative paths can be easily generated. Our future work will focus on the study of the impact of using multiple distributed computational modules on the coordination performance of multi-agent systems, and compare that of using one centralized module. Also, the implementation of generating real-time trajectory on real robotic vehicles is underway.

REFERENCES

[1] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325-344, June 2000.

[2] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376-386, June 2005.

[3] D. Hougen, S. Benjaafar, J. Bonney, J. Budenske, M. Dvorak, M. Gini, H. French, D. Krantz, P. Li, F. Malver, B. Nelson, N. Papanikolopoulos, P. Rybski, S. Stoeter, R. Voyles, and K. Yesin, "A miniature robotic system for reconnaissance and surveillance," *Proc. IEEE Int. Conf. on Robot. Autom.*, pp. 501-507, San Francisco, CA, USA, April 2000.

[4] R. R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 34, no. 2, pp. 138-153, May 2004.

[5] J. S. Jennings, G. Whelan, and W. F. Evans, Cooperative search and rescue with a team of mobile robots," *Proc. IEEE Int. Conf. Advanced Robotics*, Monterey, CA, USA, pp. 193-200, July 1997.

[6] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Trans. on Automatic Control*, vol. 38, no. 2 pp. 195-206, Feb. 1993.

[7] A. Fax and R. M. Murray, "Information flow

and cooperative control of vehicle formations," *IEEE Trans. on Automatic Control*, vol. 49, pp. 1465-1476, Sept. 2004.

[8] R. Vidal, O. Shakernia, and S. Sastry, "Formation control of nonholonomic mobile robots omnidirectional visual servoing and motion segmentation," *Proc. IEEE Conf. Robot. Autom.*, pp. 584-589, Taipei, Taiwan, Sep. 2003.

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

[10] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 813-825, Oct. 2002.

[11] P. Tabuada, G. J. Pappas, and P. Lima, "Motion feasibility of multi-agent formations," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 387-392, June 2005.

[12] T. Balch and R. Arkin, "Behavior-based formation control for multirobot systems," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926-939, Dec. 1998.

[13] J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 837-846, Oct. 2002.

[14] J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE Trans. Robot. Autom.*, vol. 19, no. 6, pp. 933-941, Dec. 2003.

[15] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 443-455, June 2004.

[16] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 905-908, Dec. 2001.

[17] P. Ogren, M. Egerstedt, and X. Hu, "A control Lyapunov function approach to multiagent coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 847-851, Oct. 2001.

[18] S. S. Ge and C.-H. Fua, "Queues and artificial potential trenches for multirobot formations," *IEEE Trans. Robot. Autom.*, vol. 21, no. 4, pp. 646-656, Aug. 2005.

[19] M. B. Milam, K. Mushambi, and R. M. Murray, "A new computational approach to real-time trajectory generation for constrained mechanical systems," *Proc. on IEEE Conf. Decision and Control*, Sydney, Australia, Dec. 2000.

[20] N. Petit, M. B. Milam, and R. M. Murray, "Inversion based constrained trajectory optimization," *Proc. IFAC Symp. Nonlinear Control Systems Design*, Saint-Petersburg, Russia, July 2001.

[21] R. Olfati-Saber, W. B. Dunbar, and R. M. Murray, "Cooperative control of multi-vehicle systems using cost graphs and optimization," *Proc. American Control Conference*, Denver, CO, USA, June 2003.

[22] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327-1360, 1995.

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

[24] 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.

[25] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[26] L. Cremean, W. B. Dunbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. M. Murray, "The Caltech multi-vehicle wireless testbed," *Proc. IEEE Conf. Decision and Control*, Las Vegas, NV, USA, pp. 86-88, Dec. 2002.

[27] F.-L. Lian and R. M. Murray, "Real-time trajectory generation for the cooperative path planning of multi-vehicle systems," *Proc. IEEE Conf. Decision and Control*, Las Vegas, NV, USA, pp. 3766-3769, Dec. 2002.

[28] F.-L. Lian and R. M. Murray, "Cooperative task planning of multi-robot systems with temporal constraints," *Proc. IEEE Int'l Conf. on Robot. Autom.*, Taipei, Taiwan, pp. 2504-2509, Sep. 2003.

Feng-Li Lian received the B.S. and M.S. degrees from National Taiwan University, Taipei, Taiwan, in 1992 and 1994, respectively, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 2001. From 2001 to 2002, he was a Postdoctoral Scholar at California Institute of Technology. In 2002, he joined the faculty of the Electrical Engineering Department, National Taiwan University, where he is currently an Associate Professor. He is the recipient of the Youth Automatic Control Engineering Award of the Chinese Automatic Control Society, Taiwan, in 2007 and the NTU Excellent Teaching Award in 2007. His current research interests include distributed and networked control systems, multiple dynamical agent systems, trajectory generation and path planning.