Paper Number: ICRA-2003

Submitted, 2003 International Conference on Robotics and Automation
http://www.cds.caltech.edu/~murray/papers/2002q_lm03-acc.html

Cooperative Task Planning of Multi-Robot Systems with Temporal Constraints[1]

Feng-Li Lian[a] and Richard Murray[b]
[(a)] Electrical Engineering, National Taiwan University
[(b)] Control and Dynamical Systems, California Institute of Technology
fengli@ccms.ntu.edu.tw, murray@caltech.edu

Abstract

This paper discusses a design methodology of cooperative trajectory generation for multi-robot systems. The trajectory of achieving cooperative tasks, i.e., with temporal constraints, is constructed by a nonlinear trajectory generation (NTG) algorithm. Three scenarios of multi-robot tasking are proposed at the cooperative task planning framework. The NTG algorithm is, then, used to generate real-time trajectory for desired robot activities. Given robot dynamics and constraints, the NTG 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-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 robots in the presence of spatial and temporal constraints. Finally, an illustrated example of cooperative task planning with temporal constraints is presented.

1 Introduction

For 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 agent systems into a hierarchical architecture is that it translates a complicate 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 complicate problem into several manageable units. The five layers and their key functionalities are (1) Network for deciding routes, (2) Link for assigning paths and target speeds, (3) Planning for managing maneuvers, (4) Regulation for completing tasks, and (5) Physical for controlling a vehicle itself. Vehicle control engineers can, then, easily and systematically specify design requirements and goals, and design different controller algorithms for each individual layer [1]. 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 (MICA) program of DARPA. 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 [2]. The layer decomposition of both AHS and MICA is 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 robot-routing case, one upper-layer controller might plan a grouping sequence of available robots and an assignment of feasible routes, and then generates an optimal activity for individual robots. 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 robot to follow. Therefore, multiple robots can utilize available resources and individually follow their own trajectories to achieve the overall system goal.

At the robot trajectory planning layer, i.e., the Regulation layer of AHS and the CPP layer of MICA, one of the challenging problems is to plan and follow a trajectory in the presence of uncertainty and limited infor-

[1]Research supported in part by DARPA MICA program, Sharon Heise, Program Manager. [a] Corresponding author: Currently with Department of Electrical Engineering, National Taiwan University, Taipei, 106, Taiwan.

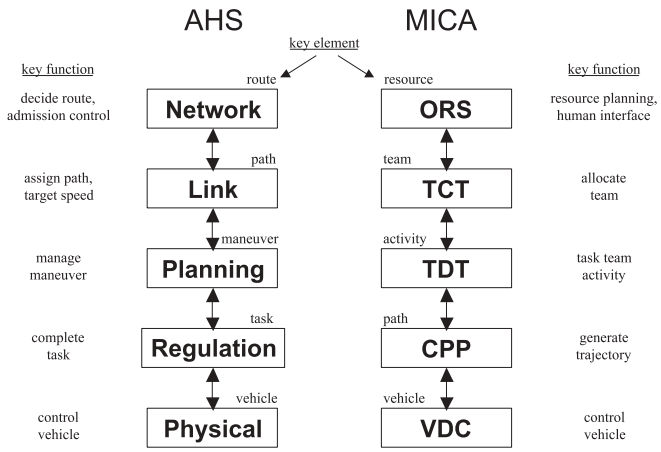


Figure 1: The AHS and MICA hierarchies and their key elements and functions.

mation. Limited information is due to the distributed nature of a multi-robot system and the range limitation of robot 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. 2, 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 robots. The proposed design architecture considers three scenarios of grouping and cooperation of multiple robots. 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 [3, 4]. In [8], we have discussed the case of only including spatial constraints in generating trajectory for the cooperative path planning of multi-vehicle systems. In order to satisfy temporal as well as spatial constraints in a cooperative multi-robot system, the NTG formulation has been further modified. Given system dynamics and state and input constraints, the NTG algorithm first finds trajectory curves in a lower dimensional space and, then, parameterizes the curves by B-splines. The coefficients of the B-splines are further solved by the sequential quadratic programming to satisfy 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

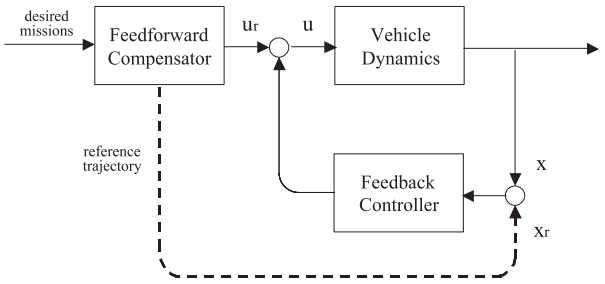


Figure 2: Two Degree of Freedom Design.

temporal constraint. The actual running time will then be recovered from the solution of the optimization approach adopted.

This paper consists of five sections, including the Introduction section. 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 temporal constraints into the NTG algorithm. Section 5 provides an illustrated example of cooperative task planning in a three-robot system. 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 as shown in Fig. 1. At the upper layer, the TCT controller plans and teams available resources such as robots 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 robots. Operator commands and environmental uncertainty as well as the constraints of teaming and activity precedence, coordinated actions, and robot dynamics are also considered at the CPP layer. Hence, the controller design at CPP is to generate cooperative trajectories of one robot or a group of robots to support the desired activities as determined by the TDT controller. In the following, three scenarios of robot activities are discussed first, and the trajectory generation algorithm will be described in the next section.

Fig. 3 shows three scenarios of robot tasking from home base (B) to target (T). In Fig. 3(a), a single robot 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-level command unit such as a TDT controller. After taking off from the home base, the robot needs to compute real-

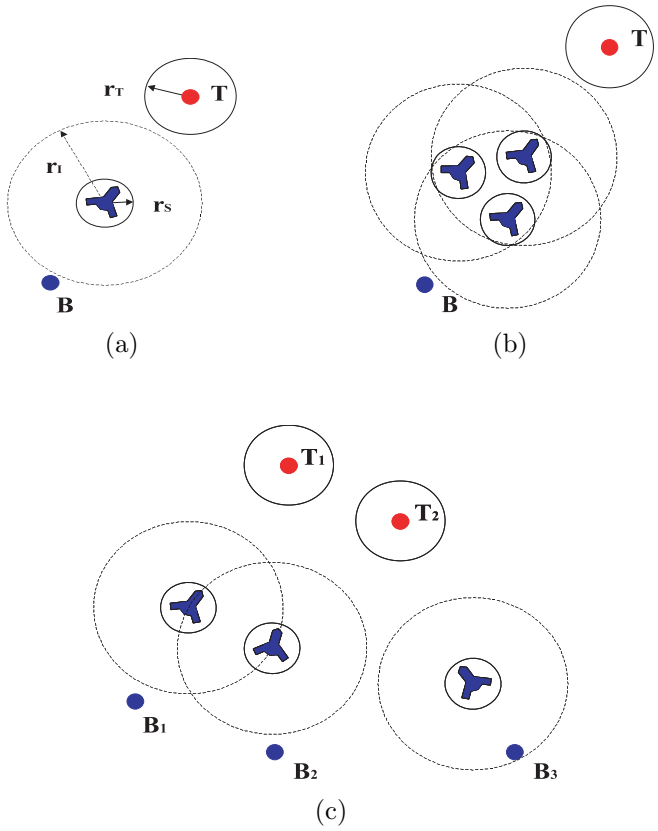


Figure 3: Three scenarios of robot tasking from home base (B) to target (T). r_S: safety radius, r_I: information radius, r_T: target detection radius.

time trajectories 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 robot 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 robot 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 robot. In general, $r_S < r_I$, otherwise, the robot 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.

The second case considers a scenario where multiple robots are commanded to accomplish the designated activity. For example, Fig. 3(b) shows that three robots are tasking from one home base to one target location. In this case, three robots might be instructed by the same activity command, and need to move together in a designated formation. Hence, the CPP controller at each individual robot should generate a set of feasible, real-time trajectories which guarantee the group of robot to move in the designated formation. A designated formation should keep the relative distance of any two robots 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 robots are moving within the adversarial area.

The third case considers a more general scenario where multiple robots from different home bases are commanded to either one common target or multiple targets. At some location, these robots 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 robot just leaves its home base, its CPP controller works like that in the first case, and, when these robots 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 robots.

In the next section, we describe the problem setup and algorithm of the NTG software package. 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 related 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 [3, 4]

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:

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

where there are N_0 initial constraints, N_f final constraints, and N_t trajectory constraints. In the robot 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:

$$\begin{aligned} J &= \phi_0(x(t_0), u(t_0)) + \phi_f(x(t_f), u(t_f)) \\ &+ \int_{t_0}^{t_f} L(x(t), u(t))dt, \end{aligned} \quad (3)$$

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:

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

such that (x, u) can be completely determined by Eq. (4), i.e.,

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

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 [5] 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:

$$(x, u) = H_1(z, z^{(1)}, ..., z^{(s_1)}), \quad \text{and} \quad (6)$$

$$0 = H_2(z, z^{(1)}, ..., z^{(s_2)}). \quad (7)$$

In this case, an additional trajectory constraint, i.e., Eq. (7), 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 curves

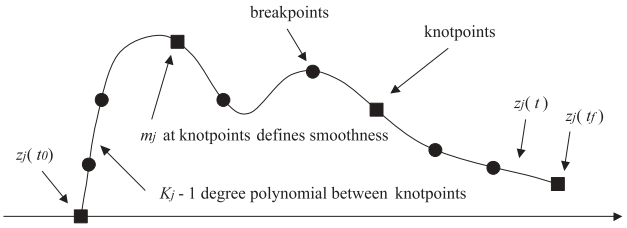


Figure 4: A B-Spline representation of z_j .

as follows [6]:

$$\begin{aligned} z_1(t) &= \sum_{i=1}^{p_1} B_{i,k_1}(t) C_i^1 \text{ for the knot sequence } \mathbf{t}_1, \\ z_2(t) &= \sum_{i=1}^{p_2} B_{i,k_2}(t) C_i^2 \text{ for the knot sequence } \mathbf{t}_2, \\ &\vdots \\ z_q(t) &= \sum_{i=1}^{p_q} B_{i,k_q}(t) C_i^q \text{ for the knot sequence } \mathbf{t}_2, \end{aligned}$$

where $B_{i,k_j}(t)$ are the B-spline basis functions for the output z_j with order k_j, C_i^j are the coefficients of the B-spline, l_j is the number of knot intervals, m_j is the number of smoothness condition at the knot point, and $p_j = l_j(k_j - m_j) + m_j$. 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 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 $\{\psi_0(\cdot,\cdot), \psi_f(\cdot,\cdot), S(\cdot,\cdot)\} \rightarrow c(y)$, 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 [7], 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 Integrating temporal constraints in NTG

According to the NTG formulation, any spatial constraints can be easily coded into the constraint set,

Eq. (2). An example of NTG for the CPP of multi-vehicle systems with spatial constraints was presented in [8]. In order to further include any temporal constraint associated to robot actions, we need to modify the original NTG formulation and augment one additional time variable into each robot dynamics [3]. We first define a new state variable T and let $T = t/\tau$, where t and τ are "old" and "new" time variables, respectively. Hence, the augmented robot dynamics becomes:

$$x' = f(x, u, T) \quad (8)$$
$$T' = 0 \quad (9)$$

where $(\cdot)' = d(\cdot)/d\tau$. Furthermore, the set of state and input constraints and additional temporal constraints can be expressed by the following set of inequalities:

$$\begin{array}{rcccl} lb_0 & \leq & \psi_0(x(0), u(0), T) & \leq & ub_0, \\ lb_f & \leq & \psi_f(x(1), u(1), T) & \leq & ub_f, \\ lb_\tau & \leq & \psi_x(x, u, T) & \leq & ub_\tau, \\ lb_T & \leq & \psi_T(T) & \leq & ub_T. \end{array} \quad (10)$$

Also, the cost function of the augmented systems can be modified as follows:

$$\begin{aligned} J &= \phi_0(x(0), u(0), T) + \phi_f(x(1), u(1), T) \\ &+ \int_0^1 L(x(\tau), u(\tau), T) d\tau. \end{aligned} \quad (11)$$

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 NTG formulation directly. The cooperative trajectory can then be generated based on different pre-specified planning time horizons of each robot activity.

In practical design situation of a multi-robot system, the number of robots could be large and the total dimension of robot dynamics 9 could be big. Also, each robot might have multiple tasks that need to be coordinated with those of other robots. Hence, the computational complexity in the robot-task space could be in the order of $n \times N \times L$, where n is the dimension of robot dynamics, N is the number of robots, and L is the number of tasks of each robot required to perform. If the total number of robots involved in the task planning is too large, the NTG algorithm might spend longer computational time to find an optimal solution. This drawback can be overcome by imposing planning time window for each robot-task, that is, adding one extra timing constraint in the fourth inequality of Eq. 10. Therefore, the task planning and trajectory generation of each robot can done separately. However, a high-level task planner is needed to generate the time window for each robot-task, and the temporal constraints

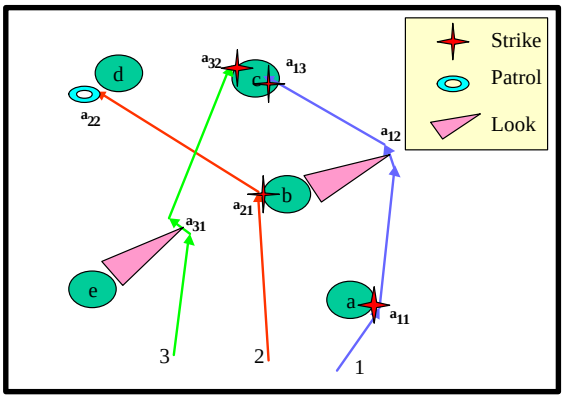


Figure 5: Activity coordination of three robots (this example is from Dr. A. Khalak of Alphatech, Inc.).

in this case will be more conservative compared with the previous case.

In next section, an illustrated example of the cooperative trajectory planning of three robots is presented.

5 Illustrative example

In this section, we use the scenario of activity coordination of different robots, to describe the integration of NTG algorithm into the MICA-CPP framework. As shown in Fig. 5, three robots are routed in a manner to achieve a set of coordinated activities (a_{ij}, denoting the jth activity of ith robot). In this example, two scenarios are considered: The Look, a_{12}, of Robot 1 on Object b must happen after the Strike, a_{21}, of Robot 2, and there is a simultaneous strike by Robots 1 and 3 on Object c. The two sets of coordinated activities can be formulated as the following temporal constraints:

$$T^{11} + T^{12} \leq T^{21} \quad (12)$$
$$T^{11} + T^{12} + T^{13} = T^{31} + T^{32}, \quad (13)$$

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

For the ease of presenting the design procedure, in this example, we consider a simplified 2-D robot dynamics described as follows:

$$\dot{x}^{ij} = u_x^{ij}, \text{ and } \dot{y}^{ij} = u_y^{ij}, i = 1, 2, 3, \quad (14)$$

where x^{ij} and y^{ij} are the coordinate of the jth activity of Robot i, and u_x^{ij} and u_y^{ij} are its corresponding inputs.

Additional state and input constraints can be further

expressed as follows:

$$\begin{array}{ccccc} r_S & \leq & \sqrt{(x^{ij}-x^{kj})^2+(y^{ij}-y^{kj})^2} & \leq & r_I, \\ u^{ij}_{lb:x,y} & \leq & u^{ij}_x, u^{ij}_y & \leq & u^{ij}_{ub:x,y}, \end{array} \quad (15)$$

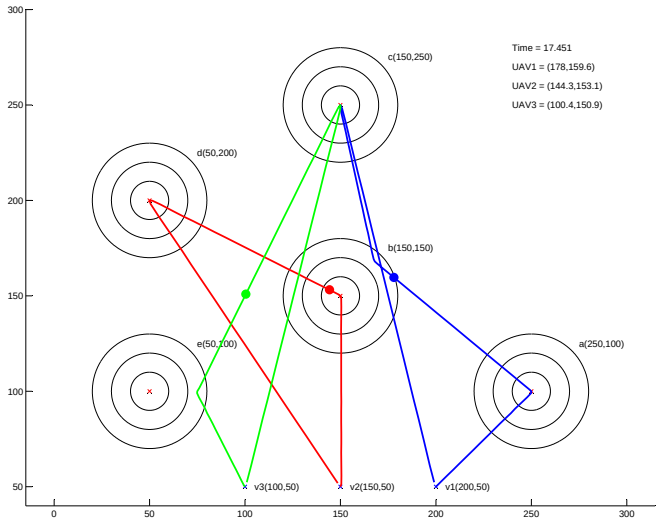
where $i,k = 1,2,3, i \neq k$, and the first inequality is for collision avoidance and obtaining information from its neighboring robots. The goal is assumed to task robots to the target by using minimal fuel and time. Hence, one choice of the cost function is as follows:

$$L(x,u) \quad = \quad \sum_{i,j} \alpha^{ij}_T (T^{ij})^2 + \alpha^{ij}_u (u^{ij}_x + u^{ij}_y)^2, \quad (16)$$

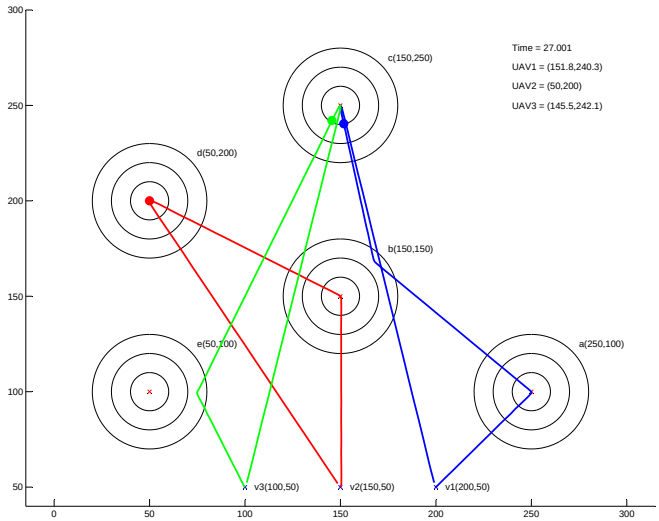
where α's are weighting factors, (x^{ij}_R, y^{ij}_R) is the reference trajectory specified by the upper-layer activity controller.

For this system, it is easy to find one set of flat outputs, z_k, such that $(x^{ij}, y^{ij}, T^{ij}, u^{ij}_x, u^{ij}_y) = (z_k, \dot{z}_k)$. 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, $z_k(t) = \sum_{i=1}^{15} B_{i,6}(t) C^k_i$ and $y = (C^k_i)$ in the nonlinear programming formulation.

The simulation result of activity coordination of three robots in two-dimensional space is shown in Fig. 6. The three robots 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. Fig. 6(a) shows the scenario that Robot 1 looks at Object b after Robot 2 strikes it and Fig. 6(b) shows the scenario that Robot 1 and Robot 3 strike Object c simultaneously.



(a)



(b)

Figure 6: Snapshots of simulation result of activity coordination of three robots.

6 Summary and future work

In this paper, we described the hierarchical design of large-scale multi-robot systems and discussed the scenario of robot tasking at the CPP layer of MICA. Based on a pre-designed robot activity, the trajectory for each robot 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. In addition to the spatial constraints, the incorporation of temporal constraints such as activity coordination was discussed in this paper. Our future work will focus on the study of the impact of using multiple distributed NTG modules on the coordination performance of multi-robot systems, and compare that of using one centralized NTG module.

References

[1] P. Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38(2):195-206, Feb. 1993.

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

[3] 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*, Sydney, Australia, Dec. 12-15, 2000.

[4] N. Petit, M. B. Milam, and R. M. Murray. Inversion based constrained trajectory optimization. *2001*

IFAC symposium on Nonlinear Control Systems Design, Saint-Petersburg, Russia, July 4-6, 2001.

[5] 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.

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

[7] 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.

[8] F.-L. Lian, and R. M. Murray. Real-Time Trajectory Generation for the Cooperative Path Planning of Multi-Vehicle Systems. *2002 Conference on Decision and Control*, Las Vegas, Nevada, Dec. 10-13, 2002.