Optimal Timing Control of Interconnected, Switched Systems with Applications to Robotic Marionettes

P. Martin and M. Egerstedt
{pmartin,magnus}@ece.gatech.edu
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

***Abstract*— We present an optimal timing control formulation of the problem of controlling autonomous puppets. In particular, by appropriately timing the different movements, entire plays can be performed. Such plays are produced by concatenating sequences of motion primitives and a compiler optimizes these sequences, using recent results in optimal switch-time control. Experimental results illustrate the operation of the proposed method.**

I. Introduction

This paper addresses the issue of when to switch between different modes of operation when controlling a dynamical system. This question falls squarely under the *optimal timing control problem* for hybrid systems, which is an area of research that has received a significant interest during the last five years. In these types of problems, the control parameter includes the schedule of the system's modes and the performance metric consists of a cost functional defined on the system's state (see [1], [4], [5], [7], [11], [16], [17], [23], [24], [25], [27]).

The work in this paper draws its inspiration from recent results on numerical optimal control of switched-mode systems, in which gradient-descent and second-order algorithms have been developed [1], [13], [28], [29]. However, what is novel in this paper as compared to the previous work is first of all the application of optimal timing control to the problem of controlling a collection of robotic puppets. Secondly, we identify a formal *motion description language* that allows us to specify what the puppets should be doing at a high level of abstraction, and then use optimal control techniques for compiling these high-level specifications into executable control code. Thirdly, a formulation of the problem is given in such a way that it can be decomposed into subproblems, each involving the optimal timing sequence for the individual puppets. The networking aspects, i.e. the way in which these subproblems are combined, are then given a direct interpretation in terms of the Lagrange multipliers in the optimization problem.

When puppeteers execute entire plays, they typically break it down into components, i.e. a play consists of multiple acts and an act consists of multiple scenes. However, within each scene, each piece, e.g. a puppet dance routine, is also broken down into components and the dance is in fact both annotated and executed as a string of movements, each of which has its own characteristics, duration, and intensity. What the research presented in this paper aims at is a way of decomposing such complex control tasks in robotics into strings of simpler control tasks, as shown in Fig. 1.

What we propose in this paper for formalizing high-level specifications for puppetry is based on Motion Description Languages (MDLs) [8], [12], [20], [21]. Specifically, a MDL is a string of pairs, each specifying what control law the system should be executing and an interrupt condition corresponding to the termination of this control law. In order for this language to be successful, it is important that it is expressive enough to be able to characterize actual puppet plays, and as such we draw inspiration from the way such plays are staged by professional puppeteers. As an example, consider a part of an actual play, as shown in Fig. 2. The play that this example comes from is the "Rainforest Adventures" - an original puppet play staged at the Center for Puppetry Arts in Atlanta during 2005 [10], [19]. It shows how the basic building blocks for a formal language for puppet choreography can be derived from existing practices in puppeteering.

In fact, the standard way in which puppet plays are described is through four parameters, namely *temporal duration*, *agent*, *space*, and *motion* (when?, who?, where?, and what?) [3], [15]. Most plays are based on *counts* in that each puppet motion is supposed to happen at a particular count. (This becomes even more important if multiple puppets are acting simultaneously on stage or if the play is set to music). At each specified count, a motion can be initiated and/or terminated.

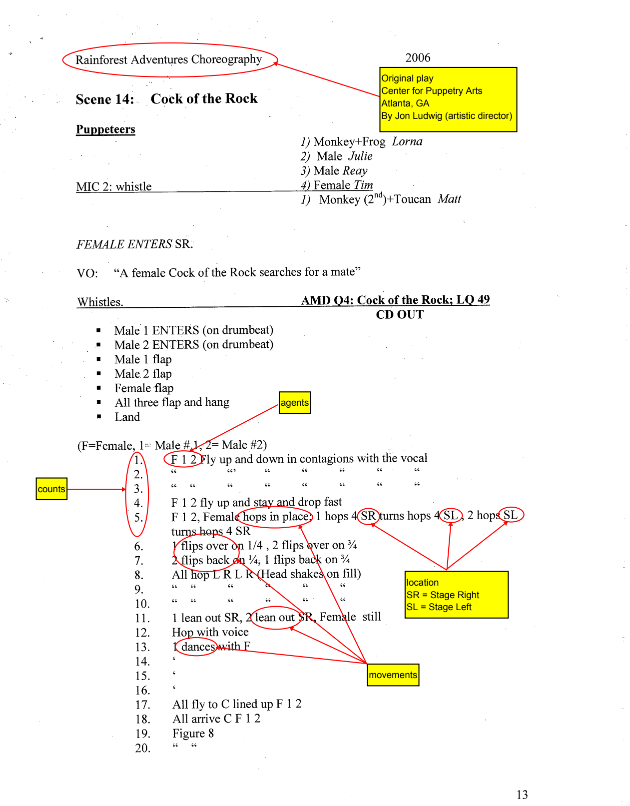
The outline of this paper is as follows: In Section II, we recall the basic definitions of a Motion Description Language and show how these definitions can be augmented to be more suitable for specifying puppet plays. We then, in Section III, use the Calculus of Variations for parsing MDL strings in an optimal way in order to produce effective control programs, as supported by experimental results. The issue of networked marionettes is the topic of Section IV, where we discuss how to structure the optimization algorithm in such a way that each puppet is (preferably) able to produce its own timing sequence without having to take into account the movements of all other puppets participating in the play.

(a) Puppet in initial configuration. (b) Puppet in wave motion. (c) Puppet starting a walk. (d) The final step in the walk mode.

Fig. 1. An image sequence of the puppet executing a *wave* followed by a *walk* mode.

II. Motion Description Languages

As the complexity of many control systems increases, due both to the system complexity (e.g. manufacturing systems, [9]) and the complexity of the environment in which the system is embedded (e.g. autonomous robots [2], [18]), multi-modal control has emerged as a useful design tool. The main idea is to define different modes of operation, e.g. with respect to a particular task, operating point, or data source. These modes are then combined according to some discrete switching logic and one attempt to formalize this notion is through the concept of a *Motion Description Language* (MDL) [8], [12], [20], [21].



Rainforest Adventures Choreography 2006

Scene 14: Cock of the Rock

Puppeteers

1) Monkey+Frog *Lorna*
2) Male *Julie*
3) Male *Reay*
4) Female *Tim*
1) Monkey (2nd)+Toucan *Matt*

MIC 2: whistle

FEMALE ENTERS SR.

VO: "A female Cock of the Rock searches for a mate"

Whistles.

AMD Q4: Cock of the Rock; LQ 49
CD OUT

- Male 1 ENTERS (on drumbeat)
- Male 2 ENTERS (on drumbeat)
- Male 1 flap
- Male 2 flap
- Female flap
- All three flap and hang
- Land

(F=Female, 1= Male #1, 2= Male #2)

1. F 1 2 Fly up and down in contagions with the vocal
2. " "" " " " " "
3. " " " " " " " "
4. F 1 2 fly up and stay and drop fast
5. F 1 2, Female hops in place; 1 hops 4 SR turns hops 4 SL, 2 hops SL turns hops 4 SR
6. 1 flips over on 1/4 , 2 flips over on ¾
7. 2 flips back on ¼, 1 flips back on ¾
8. All hop L R L R (Head shakes on fill)
9. " " " " " "
10. " " " " " "
11. 1 lean out SR, 2 lean out SR, Female still
12. Hop with voice
13. 1 dances with F
14. '
15. '
16. '
17. All fly to C lined up F 1 2
18. All arrive C F 1 2
19. Figure 8
20. " "

13

Fig. 2. Rainforest Adventures: This figure is an original puppet choreography sheet from the Center for Puppetry Arts in Atlanta [10]. It shows how the basic building blocks for a formal language for puppet choreography can be derived from existing practices in puppeteering.

Each string in a MDL corresponds to a control program composed of multiple controllers. Slightly different versions of MDLs have been proposed, but they all share the common feature that the individual atoms, concatenated together to form the control program, can be characterized by control-interrupt pairs. In other words, given a dynamical system

$$\dot{x} = f(x, u), \;\; x \in X, \;\; u \in U$$

together with a control program $(\kappa_1, \xi_1), \ldots, (\kappa_p, \xi_p)$, where $\kappa_i : X \to U$ and $\xi_i : X \to \{0, 1\}$, the system operates on this program as $\dot{x} = f(x, \kappa_1(x))$ until $\xi_1(x) = 1$. At this point the next pair is read and $\dot{x} = f(x, \kappa_2(x))$ until $\xi_2(x) = 1$, and so on. (Note that the interrupts can also be time-triggered, which can be incorporated by a simple augmentation of the state space.)

A. MDLs for Puppetry

We directly note that the general MDL outlined in the previous paragraph does not lend itself to the way puppetry plays are specified. In fact, what we will do in this section is to augment the standard MDL formulation, as discussed in [14], to include factors such as spatial location. For this, assume that the play starts at time t_0 and that it ends at time t_f. Moreover, let the temporal resolution (the length of each "count") be Δ, and assume that $(t_f - t_0)/\Delta = M$. Following this, the set of all times over which the play is specified is $\mathcal{T} = \{t_0, t_0 + \Delta, t_0 + 2\Delta, \ldots, t_0 + M\Delta\}$.

Moreover, assume that the stage is divided into N different sections (typically this number is 6, namely LowerLeft, LowerCenter, LowerRight, MiddleLeft, MiddleCenter, MiddleRight, UpperLeft, UpperCenter, UpperRight), whose planar center-of-gravity coordinates are given by $r_1, \ldots, r_N$, with the set of regions being given by $\mathcal{R} = \{r_1, \ldots, r_N\}$.

As before, assume that the puppet under consideration has the dynamics

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

Now, given that we have constructed a number of control laws $\kappa_j,\ j = 1\ldots, C$, corresponding to different moves that the puppet can perform, with each control law being a function of x (state), t (time), and α (a parameter characterizing certain aspects of the motion such as speed, energy, or acceleration, as is the normal interpretation of the parameterization of biological motor programs), we can let the set of moves that puppet can perform be given by $\mathcal{K} = \{\kappa_1, \ldots, \kappa_C\}$. In fact, we will often use the shorthand $f_j(x, t, \alpha)$ to denote the impact that control law κ_j has through $f(x, \kappa_j(x, t, \alpha))$.

As already pointed out, each instruction in the puppet play language is a four-tuple designating *when*, *who*, *where*, and *what* the puppets should be doing. In other words, we let the motion alphabet be given by $\mathcal{L} = \mathcal{T} \times \mathcal{T} \times \mathcal{R} \times \mathcal{K}$. Each element in $\mathcal{L}$ is thus given by (T_0, T_1, r, κ), where the interpretation is that the motion should take place during the time interval $T_1 - T_0$, largely in region r, while executing the control law κ.

Following the standard notation in the formal language field, we let $\mathcal{L}^\star$ denote the set of all finite-length concatenations of elements in $\mathcal{L}$ (including the empty string), and let puppet plays be given by words $\lambda \in \mathcal{L}^\star$. In particular, if we let $\lambda = (t_0, T_1, r_1, \kappa_1), (T_1, T_2, r_2, \kappa_2), \ldots, (T_{p-1}, T_p, r_p, \kappa_p)$, then the puppet operates on this string through

$$\dot{x} = \begin{cases} f_1(x, t, \alpha_1),\ t \in [t_0, T_1) \\ f_2(x, t, \alpha_2),\ t \in [T_1, T_2) \\ \quad\vdots \\ f_p(x, t, \alpha_p),\ t \in [T_{p-1}, T_p]. \end{cases}$$

This seems fairly natural, but two essential parameters have been left out. First, the motion parameters $\alpha_1, \ldots, \alpha_p$ have not yet been specified. Moreover, the desired regions $r_1, \ldots, r_p$ have not been utilized in any way. In order to remedy this, we need to construct not just a *parser* for puppet plays, as given above, but also a *compiler* that selects the "best" parameters (as well as durations) for the different moves so that the play is executed as efficiently as possible, which is the topic of the next section.

III. Compiling MDL Strings Through Optimal Control

In this section we present a compiler that takes as inputs strings in a MDL and optimizes over these strings by adjusting the interrupt times as well as the parameters defining the specifics of the individual control laws. Rather than solving a large-scale problem explicitly, we start with the canonical two-primitive MDL string. In fact, consider the following optimal control problem:

$$\min_{\tau,\alpha_1,\alpha_2} J(\tau, \alpha_1, \alpha_2) = \int_0^{t_f} L(x, t)dt + C_1(\alpha_1) + C_2(\alpha_2) + D(\tau) + \Psi_1(x(\tau)) + \Psi_2(x(t_f)),$$

where

$$\dot{x} = \begin{cases} f_1(x, t, \alpha_1),\ t \in [0, \tau) \\ f_2(x, t, \alpha_2),\ t \in [\tau, t_f] \end{cases}$$
$$x(0) = x_0.$$

This optimal control problem is the atomic problem involving how to execute the two-instruction play $(0, T, r_1, \kappa_1), (T, t_f, r_2, \kappa_2)$ under the interpretation that $D(\tau)$: is a cost that penalizes deviations from the presepcified, nominal switching time T, $C_i(\alpha_i)$ measures how much energy it takes to use parameter α_i for mode i, $\Psi_i(\cdot)$: ensures that the puppet is close to r_1 at time τ (and similarly for $x(t_f)$), and $L(x, t)$ is a trajectory cost that may be used to ensure that a reference trajectory is followed.

We can apply calculus of variations techniques, under suitable assumptions of continuous differentiability, to the cost functional. The derivations are straightforward and follow those of [1], [13], and they result in the optimality conditions

$$\begin{aligned} \frac{\partial J}{\partial \tau} &= \lambda(\tau_-)f_1(x(\tau)) - \lambda(\tau_+)f_2(x(\tau)) + \frac{\partial D}{\partial \tau} \\ \frac{\partial J}{\partial \alpha_2} &= \mu(\tau_+) \\ \frac{\partial J}{\partial \alpha_1} &= \mu(0), \end{aligned}$$

where the co-states λ and μ satisfy the following discontinuous (backwards) differential equations:

$$\begin{aligned} \lambda(T) &= \frac{\partial \Psi_2}{\partial x}(x(T)) \\ \dot{\lambda} &= -\frac{\partial L}{\partial x} - \lambda\frac{\partial f_2}{\partial x},\ t \in (\tau, T) \\ \lambda(\tau_-) &= \lambda(\tau_+) + \frac{\partial \Psi_1}{\partial x}(x(\tau)) \\ \dot{\lambda} &= -\frac{\partial L}{\partial x} - \lambda\frac{\partial f_1}{\partial x},\ t \in [0, \tau) \\ \mu(T) &= \frac{\partial C_2}{\partial \alpha_2} \\ \dot{\mu} &= -\lambda\frac{\partial f_2}{\partial x},\ t \in (\tau, T) \\ \mu(\tau_-) &= \frac{\partial C_1}{\partial \alpha_1} \\ \dot{\mu} &= \lambda\frac{\partial f_1}{\partial x},\ t \in [0, \tau). \end{aligned}$$

By a direct generalization to more than two modes, this construction allows us to produce a compiler that takes *plays* and outputs *strings of control modes with an optimized temporal duration and mode parameterization*.

A. Example

As an illustrative example, consider the following cost functional:

$$J(\tau, \alpha_1, \alpha_2) = \int_0^{t_f} q^T Pq\ dt + \rho(T - \tau)^2 + w_1\alpha_1^2 + w_2\alpha_2^2,$$

where $q \in \mathbb{R}^6$ is the vector of generalized joint angles of the puppet, P is a suitable weight matrix, and ρ, w_1, w_2 are cost weights that prescribe relative weights to deviations from the nominal switch time and motion intensity parameters.

In Figure 3, the joint angles corresponding to the initial conditions $\tau = 4$, $\alpha_1 = \alpha_2 = 1$ are shown. Note that this initial trajectory results in the left arm's odd looking behavior, where the wave motion stops the left arm in "mid-air" as the walk motion begins. A more desirable trajectory would lower the left arm completely before initiating a walk. Therefore, we defined the weight matrix, P, such that the joint angles are penalized for deviations from the puppet's "home" position, i.e. the left arm initial joint angles $\theta_l = \phi_l = 0$. After an iterative, descent-based optimization algorithm has terminated, the new values become $\tau = 4.3423$, $\alpha_1 = 1.3657$, $\alpha_2 = 0.9566$, with the corresponding joint angles shown in Figure 4. This plot shows that at termination the joint angles were close to 0 before starting the walk mode, resulting in a more natural looking motion.

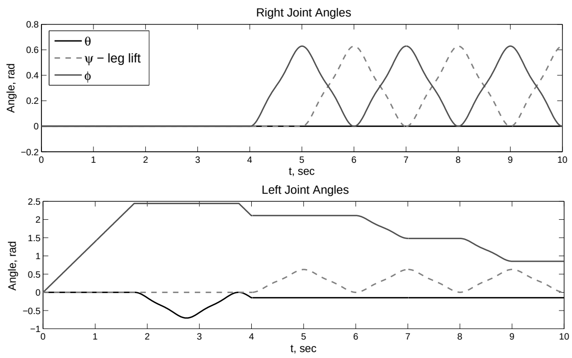


Fig. 3. Original joint angle trajectories.

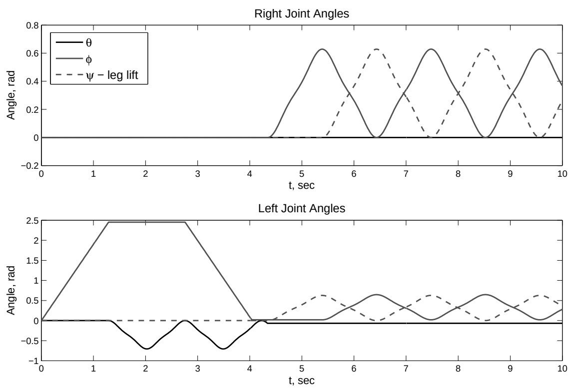
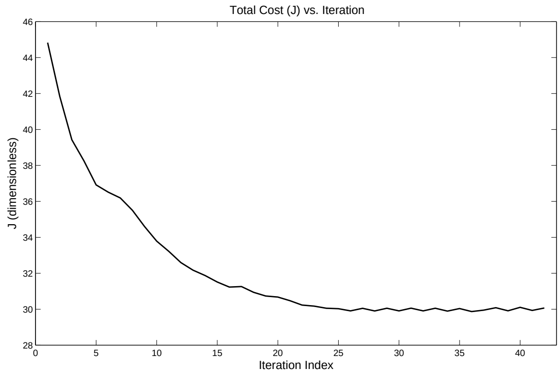


Fig. 4. Optimized joint angle trajectories.

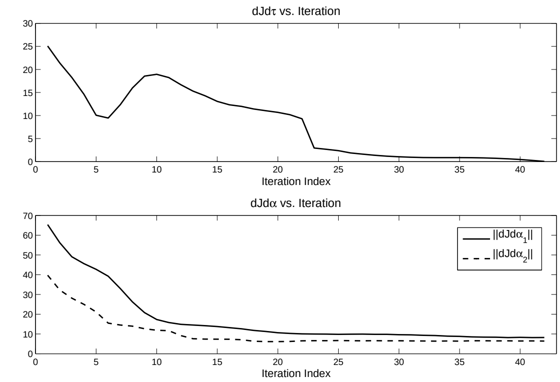
Examining the total cost plot in Fig. 5(a) reveals that the cost is indeed appropriately reduced as the gradient descent algorithm ran past 20 iterations. Additionally, the derivatives of $J(\tau, \alpha_1, \alpha_2)$ in Fig. 5(b) are shown to decrease over time, and given enough iterations approaches 0.

B. Experimental Platform

In conjunction with the simulation results above, we have also developed a hardware platform, as shown in Fig. 1. In fact, the movement in that figure is the one obtain in the



(a) Total cost as a function of iteration.



(b) Values of $\frac{\partial J}{\partial \tau}$, $\frac{\partial J}{\partial \alpha_1}$, and $\frac{\partial J}{\partial \alpha_2}$

Fig. 5. Plots of the total cost and the value of its various derivatives over simulation time.

previous section, in which the puppet switches between a wave and a walk mode.

The puppet system is comprised of three components: hardware system, a Java control application, and Matlab optimization routines. A diagram of the architecture is seen in Fig. 6.

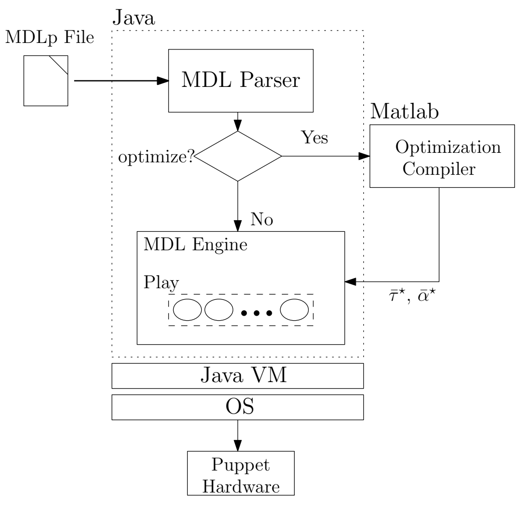


Fig. 6. The puppet system software architecture.

The puppet hardware is controlled by six Robotis Dynamixel motors [26]. These motors are suitable for this application since they can be issued position *and* velocity commands. Additionally, they are linked together by a multipoint serial interface (the RS-485 standard), which enables a single command to generate motion on the six motors simultaneously. A microcontroller connects all of the motors together and communicates with a host computer via a serial port.

The Java application has several functional pieces. First, it manages the serial port so that commands may be sent to the underlying puppet hardware. Additionally, it can parse MDL files and generate a string of modes based on what is available in the MDL library. These modes are then fed into an *MDL engine*, which applies a particular mode's control action to generate a motion command. Once a user has constructed a play, it may be fed into a Matlab optimization routine, where the algorithm of section III and the kinematics of the puppet are implemented. The Matlab routine then outputs the optimal switch times ($\bar{\tau}^\star$) and scaling factors ($\bar{\alpha}^\star$).

IV. Networked Timing Control

Since most interesting puppet plays include more than one marionette, we will be forced to handle situations in which two (or more) puppets need to execute a movement in a coordinated fashion. Due to the risk of tangling strings, multi-puppet coordination is mainly done through spatial and temporal adjacency.

For this, we assume that the play is comprised of n puppets, each operating under their own dynamics. Additionally, each puppet switches between m_i control modes, with (as before) the terminal time denoted by $T = \tau_{m_i},\ i = 1, \ldots, n$.

In other words, a direct modification to the previous formulation gives that each puppet be governed by the dynamics,

$$\dot{x}_i(t) = \begin{cases} f_{i,1}(x,t),\ t \in [0, \tau_{i,1}) \\ f_{i,2}(x,t),\ t \in [\tau_{i,1}, \tau_{i,2}) \\ \vdots \\ f_{i,m_i}(x,t),\ t \in [\tau_{i,m_i-1}, \tau_{i,m_i}] \end{cases}$$

for $i = 1, \cdots, n$. Let moreover the cost functional be defined as

$$J(\bar{\tau}_1, \ldots, \bar{\tau}_n) = \int_0^T \sum_{i=1}^n D_i(x_i, t)\ dt = \sum_{i=1}^n J_i(\bar{\tau}_i)$$

where $D_i(x,t)$ is the cost associated with operating system i for $t \in [0, \tau_{i,m})$.

Now, to illustrate the way in which the temporal constraints show up, we, for the ease of notation (but without loss of generality) assume that the temporal constraint only affects the d^{th} switch for systems j and k, where $j, k \in \{1, \cdots, n\}$, as $c_d(\tau_{j,d}, \tau_{k,d}) = \tau_{j,d} - \tau_{k,d} \leqslant 0$.

It is directly clear that the way this optimization problem can be solved is by simply augmenting the cost with the Lagrangian term $\mu c_d(\tau_{j,d}, \tau_{k,d})$, and then solve the problem jointly across all the switching times for all the puppets. However, we do not want to do this, and we instead illustrate how recent ideas from so-called *Team Theory*, as described in [22], can help distribute the computational burden across the different puppets. (Note that the details given below are not due to us, but rather that we highlight their application to the problem of distributed timing control as it pertains to the robotic marionette application.)

A. Distributed Coordination

Let, as before, puppets j and k ($j \neq k$) be temporally constrained via the d^{th} switch as $c_d(\tau_{j,d}, \tau_{k,d}) \leq 0$. Using the developments in [22], the constrained problem becomes

$$L(\tau_{j,d}, \tau_{k,d}, \mu) = J_j(\tau_{j,d}) + J_k(\tau_{k,d}) + \mu c_d(\tau_{j,d}, \tau_{k,d})$$

where we have assumed (without loss of generality) that the only control parameters are $\tau_{j,d}$ and $\tau_{k,d}$. It should directly be noted that the cost functionals are decoupled (i.e. cost J_j depends *only* on system j's dynamics). Therefore, taking the derivative of the Lagrangian with respect to μ results in the expression,

$$\frac{\partial L}{\partial \mu} = \tau_{j,d} - \tau_{k,d},$$

in combination with the previously defined gradient expressions defined with respect to the switching times.

Now, algorithmically, this formulation is interesting in that the dual problem becomes $g^\star = \max_\mu g(\mu),\ \mu \geqslant 0$, where

$$g(\mu) = \inf_{\tau_{j,d}, \tau_{k,d}} \{J_j(\tau_{j,d}) + J_k(\tau_{k,d}) + \mu(\tau_{j,d} - \tau_{k,d})\}.$$

As such, using a gradient descent for the switch times, and a gradient ascent for the Lagrange multiplier μ, allows us to largely decouple the solution and let the networking aspect be reflected only through the update of the multiplier, as was done in [22]. In fact, if we let

$$\begin{aligned} \dot{\tau}_{j,d} &= -\tfrac{\partial J_j}{\partial \tau_{j,d}} - \mu \\ \dot{\tau}_{k,d} &= -\tfrac{\partial J_k}{\partial \tau_{k,d}} + \mu \\ \dot{\mu} &= \tau_{j,d} - \tau_{k,d} \end{aligned}$$

all that needs to be propagated between the two systems is the value of the Lagrange multiplier μ. This observation, developed in [22], thus leads us to a general architecture for solving networked switching time optimization problems, as shown in Figure 7. A more complete exploration of this issue is, however, left to a future endeavor.

V. Conclusions

In this paper we presented the a motion description language for specifying and encoding autonomous puppetry plays in a manner that is faithful to standard puppetry choreography. The resulting strings of control-interrupt pairs are then compiled in the sense that they are parsed by a dynamical system that produces optimized, hybrid control laws corresponding to strings of motions, locations, and temporal durations for each motion primitive. This paper also discusses some issues arising in modeling and how to capture relevant motion primitives from empirical data. Experimental and simulation results illustrate the viability of the proposed approach.

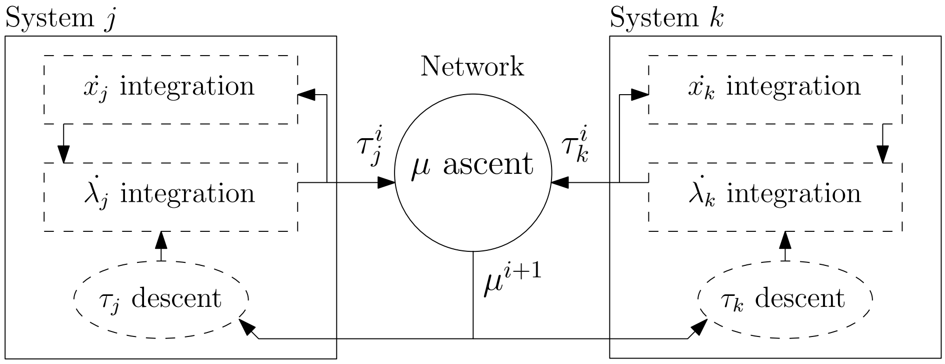


Fig. 7. This figure shows how to propagate information between the two subsystems (puppets) in order to solve the networked timing problem.

References

[1] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest. A Gradient Descent Approach to Optimal Mode Scheduling in Hybrid Dynamical Systems. *Journal of Optimization Theory and Applications.* To appear 2008.

[2] R.C. Arkin. *Behavior Based Robotics.* The MIT Press, Cambridge, MA, 1998.

[3] B. Baird. *The Art of the Puppet.* Mcmillan Company, New York, 1965.

[4] A. Bemporad, F. Borrelli, and M. Morari. Piecewise Linear Optimal Controllers for Hybrid Systems. In *Proc. of the American Control Conf.*, pp. 1190-1194, 2000.

[5] A. Bemporad, F. Borrelli, and M. Morari. On the Optimal Control Law for Linear Discrete Time Hybrid Systems. In *Hybrid Systems: Computation and Control*, M. Greenstreet and C. Tomlin, Editors, Springer-Verlag LNCS 2289, pp. 105-119, 2002.

[6] Blender. *http://www.blender.org*, 2007.

[7] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A Unified Framework for Hybrid Control: Model and Optimal Control Theory. *IEEE Trans. on Automatic Control*, Vol. 43, pp. 31-45, 1998.

[8] R.W. Brockett. On the Computer Control of Movement. In the *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, pp. 534–540, New York, April 1988.

[9] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, Norwell, MA, 1999.

[10] Center for Puppetry Arts. http://www.puppet.org/.

[11] J. Chudoung and C. Beck. The Minimum Principle for Deterministic Impulsive Control Systems. *IEEE Conf. on Decision and Control*, pp. 3569–3574, 2001.

[12] M. Egerstedt and R.W. Brockett. Feedback Can Reduce the Specification Complexity of Motor Programs. *IEEE Transactions on Automatic Control*, Vol. 48, No. 2, pp. 213–223, Feb. 2003.

[13] M. Egerstedt, Y. Wardi, and H. Axelsson. Transition-Time Optimization for Switched-Mode Dynamical Systems. *IEEE Trans. on Automatic Control*, Vol. AC-51, pp. 110-115, 2006.

[14] M. Egerstedt, T.Murphey, and J. Ludwig. Motion Programs for Puppet Choreography and Control. *Hybrid Systems: Computation and Control*, Springer-Verlag, pp. 190-202, Pisa, Italy April 2007.

[15] L. Engler and C. Fijan. *Making Puppets Come Alive*. Taplinger Publishing Company, New York, 1973.

[16] A. Guia, C. Seatzu, and C. Van der Mee. Optimal Control of Switched Autonomous Linear Systems. In *IEEE Conf. on Decision and Control*, pp. 1816-1821, 1999.

[17] S. Hedlund and A. Rantzer. Optimal Control of Hybrid Systems. *IEEE Conf. on Decision and Control*, pp. 3972-3977, 1999.

[18] D. Kortenkamp, R.P. Bonasso, and R. Murphy, Eds. *Artificial Intelligence and Mobile Robots*. The MIT Press, Cambridge, MA, 1998.

[19] J. Ludwig. Rainforest adventures. http://www.puppet.org/perform/rainforest.shtml.

[20] D. Hristu-Varsakelis, M. Egerstedt, and P.S. Krishnaprasad. On The Structural Complexity of the Motion Description Language MDLe. *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

[21] V. Manikonda, P.S. Krishnaprasad, and J. Hendler. Languages, Behaviors, Hybrid Architectures and Motion Control. In *Mathematical Control Theory*, Eds. Willems and Baillieul, pp. 199–226, Springer-Verlag, 1998.

[22] A. Rantzer. On Price Mechnanisms in Linear Quadratic Team Theory. *IEEE Conference on Decision and Control*, New Orleans, LA, Dec. 2007.

[23] M.S. Shaikh and P. Caines. On Trajectory Optimization for Hybrid Systems: Theory and Algorithms for Fixed Schedules. *IEEE Conf. on Decision and Control*, pp. 1997–1998, 2002.

[24] M.S. Shaikh and P.E. Caines. On the Optimal Control of Hybrid Systems: Optimization of Trajectories, Switching Times and Location Schedules. In *6th International Workshop on Hybrid Systems: Computation and Control*, 2003.

[25] H.J. Sussmann. Set-Valued Differentials and the Hybrid Maximum Principle. *IEEE Conf. on Decision and Control*, pp. 558–563, 2000.

[26] Tribotix. *http://www.tribotix.com*, 2007.

[27] L.Y. Wang, A. Beydoun, J. Cook, J. Sun, and I. Kolmanovsky. Optimal Hybrid Control with Applications to Automotive Powertrain Systems. *Control Using Logic-Based Switching*, Vol. 222 of LNCIS, pp. 190-200, Springer-Verlag, 1997.

[28] X. Xu and P. Antsaklis. Optimal Control of Switched Autonomous Systems. *IEEE Conf. on Decision and Control*, pp. 4401–4406, 2002.

[29] X. Xu and P.J. Antsaklis. Optimal Control of Switched Systems via Nonlinear Optimization Based on Direct Differentiations of Value Functions. *Int. J. of Control*, Vol. 75, pp. 1406-1426, 2002.