Motion Programs for Puppet Choreography and Control

Magnus Egerstedt[1], Todd Murphey[2], and Jon Ludwig[3]

[1] Georgia Institute of Technology, Electrical and Computer Engineering
Atlanta, GA 30323, USA
magnus@ece.gatech.edu

[2] University of Colorado, Electrical and Computer Engineering
Boulder, CO 80309, USA
murphey@colorado.edu

[3] The Center for Puppetry Arts, Atlanta, GA 30309, USA
jonludwig@puppet.org

Abstract. This paper presents a motion description language (MDLp) for specifying and encoding autonomous puppetry plays in a manner that is faithful to the way puppetry choreography is currently formulated. In particular, MDLp is a formal language whose strings, when parsed by a dynamical system (the puppet) produces optimized, hybrid control laws corresponding to strings of motions, locations, and temporal durations for each agent. The paper is concerned with the development of this language as well as with an optimization engine for hybrid optimal control of MDLp strings, and with the generation of motion primitives within the "Imitate, Simplify, Exaggerate" puppetry paradigm.

1 Introduction

One of the main drivers behind the rapidly emerging abstraction-based approach to control and software design is the ability to specify the desired system behavior at a high-level of abstraction, without having to take the actual implementation details into account [9,21]. In other words, the key idea is to be able to give high-level specifications in some language such as linear temporal logic (LTL) [17,22], Computation and Control Languages (CCL) [16], maneuver automata [13], or Motion Description Languages (MDL) [4,10,14,20], and then be assured that the transitions from high-level specifications to actual control signals are achieved in a stable and correct manner.

In this paper we pursue this issue of abstraction-based control for a particular application, namely *autonomous puppetry*, which apart from being a conceptual oxymoron, presents a number of technical challenges. The ultimate objective is to be able to input high-level descriptions of desired puppet motions, denoted *plays*, and then go from such plays to actual control laws for implementing the plays on autonomous marionette puppets, as shown in Figure 1.

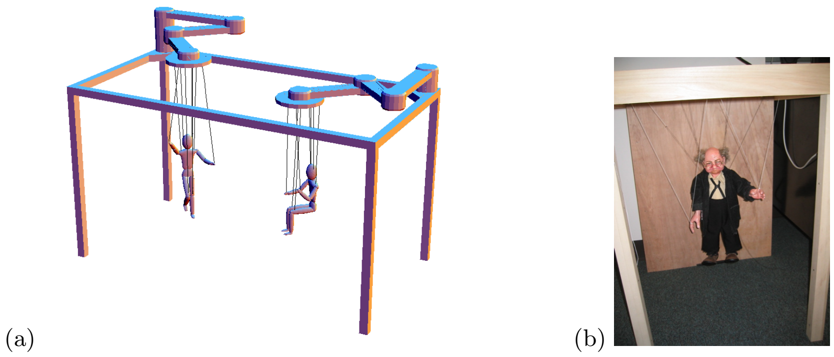


Fig. 1. A *Mathematica* graphical representation of the mechanical system for autonomous puppetry control (a), together with one of the marionettes used (b)

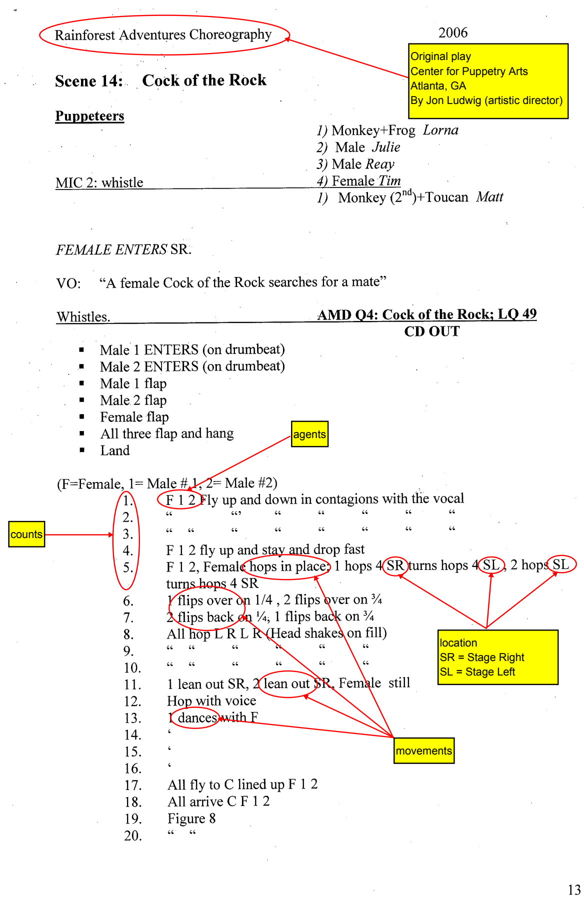
The control strategy that we will employ will be hybrid, for three distinct reasons, namely:

1. Plays are naturally described as sequences of distinctive motions, which means that the controller must switch between different modes of operation;
2. As the objective is to mimic human (or animal) behaviors, puppeteers typically work with a set of established motion primitives, such as "walk", "run", "dance", "hop" [11]; and
3. The marionette platform is in itself hybrid in that the strings (actuation modalities) are in a number of different configurations during a play, including "free", "controlled", "locked", or "grouped" [15].

We will discuss these three hybrid aspects of the autonomous marionette project, and the framework that we propose in this paper for formalizing high-level specifications for puppetry is based Motion Description Languages. 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. The particular language that we propose is slightly more structured than the standard MDL (or MDLe, where "e" stands for "extended") and we will call this language MDLp, with "p" meaning "puppetry". 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 Figure 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 [7,19]. 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?,


Original play
Center for Puppetry Arts
Atlanta, GA
By Jon Ludwig (artistic director)
agents
counts
location
SR = Stage Right
SL = Stage Left
movements

Rainforest Adventures Choreography 2006

Scene 14: Cock of the Rock

Puppeteers

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

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 [7]. It shows how the basic building blocks for a formal language for puppet choreography can be derived from existing practices in puppeteering.

who?, where?, and what?) [2,11]. 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 is initiated and/or terminated. Following this standard practice, we, in the following sections, will propose a formal language for describing such puppet plays, and the outline of this paper is as follows: In Section 2, we recall the basic definitions of a Motion Description Language and show how these definitions can be augmented to form the MDLp, suitable for specifying puppet plays. We then, in Section 3, use the Calculus of Variations for parsing MDLp strings in an optimal way in order to produce effective control programs, deployable on the robot platform. The question of how to generate the motion primitives that constitute the building blocks of MDLp is the focus of Section 4. In fact, professional puppeteers use an expression, “Imitate, Simplify, Exaggerate”, to describe the basic steps in making a puppet perform a given behavior [11]. First, imitate the behavior that one observes, then simplify it down to its basic components, and finally exaggerate the resulting behavior to convey the correct level of animation and emotional content to the viewer, who is often quite distant from the stage. These three steps have formal mathematical counterparts, which is the focus of Section 4, followed by the conclusions, in Section 5.

2 Choreography

2.1 Motion Description Languages

As the complexity of many control systems increases, due both to the system complexity (e.g. manufacturing systems, [6]) and the complexity of the environment in which the system is embedded (e.g. autonomous robots [1,18]), 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) [4,10,14,20].

Each string in a MDL corresponds to a control program that can be operated on by the control system. 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

$$\begin{aligned}&\dot{x} = f(x,u),\ \ x \in \mathbb{R}^N,\ \ u \in U\\&y = h(x),\ \ y \in Y,\end{aligned}$$

together with a control program $(k_1, \xi_1), \ldots, (k_z, \xi_z)$, where $k_i\ :\ Y \to U$ and $\xi_i\ :\ Y \to \{0,1\}$, the system operates on this program as $\dot{x} = f(x, k_1(h(x)))$ until $\xi_1(h(x)) = 1$. At this point the next pair is read and $\dot{x} = f(x, k_2(h(x)))$ until $\xi_2(h(x)) = 1$, 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 number of results have been derived for such (and similar) systems, driven by strings of symbolic inputs. For example, in [3], the set of reachable states was characterized, while [12] investigated optimal control aspects of such systems. In [8,14,20], the connection between MDLs and robotics was investigated.

2.2 Puppet Dynamics

Before we can establish a suitable formalism for motion control of autonomous marionettes, an appropriate puppet model is needed. In fact, we let the puppet be modeled using well-known Euler-Lagrange methods for articulated mechanical systems [5]. Our puppet system is similar to a closed-chain of rigid bodies, but differs in that one of the linkages has no mass. This implies that one cannot apply a force to the link or use the inertia of the link. In fact, by including the parameters that define the string link (e.g., the location of the string end-points) in the configuration, we get a globally degenerate inertia matrix and, correspondingly, degenerate equations of motion. Associating a mass with the string parameters to avoid this difficulty introduces other problems in that a small mass yields stiff differential equations for the motion while a large mass will unrealistically affect the dynamics of the system.

The solution is to treat the string as a constraint that indirectly applies to the system inputs. Hence, we will treat the mechanical superstructure as a kinematic system while we treat the puppet itself as a dynamic system. Thus, we are modeling the puppet using a mixed dynamic-kinematic model.

The validity of such a partial kinematic reduction can be verified as follows (under the assumption that the mechanical system controlling the puppet is fully-actuated). The mechanical system, including both the puppet and the mechanism controlling the puppet, can be described using the constrained Euler-Lagrange equations with $q = [q_D, q_K, q_M]$, where q_M describes the configuration of the mechanism, q_K describes the configuration of the strings, and q_D describes the configuration of the puppet. The equations of motion can be written as: $\tilde{\nabla}_{\dot{q}}\dot{q} = u_i Y^i$ where $\tilde{\nabla}$ is the constrained affine connection, u^i are the m inputs, and Y_i are the associated input vector fields. (See [5] for a complete description of this formalism.) In this context, a system is *kinematically reducible* (i.e, all paths on the configuration manifold Q correspond to trajectories on TQ and vice-versa) if $\langle Y_i, Y_j \rangle \in span\{Y_k \mid 1 \leq k \leq m\}$ where $\langle X, Y \rangle = \tilde{\nabla}_X Y + \tilde{\nabla}_Y X$. Now, because the strings are massless, the inertia tensor is block diagonal in $[q_D, q_K]$. This allows us to address the constraints independently as constraints between q_D and q_M so that

$$\begin{aligned} \tilde{\nabla}_{\dot{q}_M}\dot{q}_M &= u_i Y^i \\ \tilde{\nabla}_{\dot{q}_D}\dot{q}_D &= 0. \end{aligned}$$

Because the mechanism controlling the puppet is *assumed* to be fully actuated, the first equation for the mechanism dynamics trivially satisfies the condition $\langle Y_i, Y_j \rangle \in span\{Y_k \mid 1 \leq k \leq m\}$ for kinematic reducibility. Hence, separating the kinematic reduction of the mechanism controlling the puppet from the (*not* kinematically reducible) dynamics of the puppet is a mechanically valid description of the system.

Lastly, the string constraints are actually inequality constraints. In fact, the strings can go slack. This can be included by monitoring the Lagrange multipliers enforcing the constraints and using projection operators to provide impulses that release the constraint when the string goes slack and enforce them again when the end of the string is reached.

2.3 MDLp

Now that we have a model of the puppet dynamics, we note that the general MDL outlined in Section 2.1 does not lend itself to be directly applicable to the scenario described in Figure 2. In fact, what we will do in this section is to augment the standard MDL formulation 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 $\mathcal{R} = \{r_1, \ldots, r_N\}$.

From the arguments in Section 2.2, we can assume that each puppet has a dynamics given by

$$\dot{x}^i = f^i(x^i, u^i),\ y^i = \pi^i(x^i),$$

where the superscript i denotes agent i, and the output $y^i \in \mathbb{R}^2$ is given by a projection π^i from X^i to the plane. Now, given that we have constructed a number of control laws $\kappa^i_j,\ j = 1\ldots, C^i$, corresponding to different moves that puppet i can perform, with each control law being a function of x^i (state), t (time), and α_i (a parameter characterizing certain aspects of the motion such as speed, energy, or acceleration, as is the normal interpretation of the parametrization of biological motor programs), we can let the set of moves that puppet i can perform be given by $\mathcal{K}^i = \{\kappa^i_1, \ldots, \kappa^i_{C^i}\}$. In fact, we will often use the shorthand $f^i_j(x^i, t, \alpha)$ to denote the impact that control law κ^i_j has on puppet i through $f^i(x^i, \kappa^i_j(x^i, t, \alpha))$.

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 associated with puppet i be given by $\mathcal{L}^i = \mathcal{T} \times \mathcal{T} \times \mathcal{R} \times \mathcal{K}^i$. Each element in $\mathcal{L}^i$ 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.

For the reminder of this section, we will drop the explicit dependence on i, and assume that we are concerned with a given, individual puppet. We can then follow the standard notation in the formal language field and 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.

3 A Puppet Play Compiler

Consider the following optimal control problem:

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

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 following interpretations

$D(\tau) =$ function that penalizes deviations from$T$, e.g. $(\tau - T)^2$
$C_i(\alpha_i) =$function that measures how much energy it takes to use parameter α_i
$\Psi_i(x(\tau \text{ or } T)) =$ function that ensures that the projection $\pi((x(\tau))$ is close to $r_1$and similarly for $\pi(x(T))$
$L(x,t) =$ function that may be used to ensure that a reference trajectory is followed.

By forming the Lagrangian

$$\tilde{J} = \int_0^{\tau} (L + \lambda_1(f_1 - \dot{x}))dt + \int_{\tau}^{T} (L + \lambda_2(f_2 - \dot{x}))dt + C_1 + C_2 + D + \Psi_1 + \Psi_2$$

and using a standard variational argument, with $\tau \to \tau + \epsilon\theta,\ \alpha_1 \to \alpha + \epsilon a_1,\ \alpha_2 \to \alpha_2 + \epsilon a_2$, we can obtain the corresponding variation in the trajectory as $x(t) \to x(t) + \epsilon\eta(t)$, with $\eta(0) = 0$.

By letting $\tilde{J}_\epsilon$ denote the Lagrangian from the variational system, we get that

$$\begin{aligned}
\lim_{\epsilon\to 0} \frac{\tilde{J}_\epsilon - \tilde{J}}{\epsilon} = & \int_0^{\tau} \left(\left(\frac{\partial L}{\partial x} + \lambda_1 \frac{\partial f_1}{\partial x} + \dot{\lambda}_1 \right) \eta + \lambda_1 \frac{\partial f_1}{\partial \alpha_1} a_1 \right) dt \\
& + \int_{\tau}^{T} \left(\left(\frac{\partial L}{\partial x} + \lambda_2 \frac{\partial f_2}{\partial x} + \dot{\lambda}_2 \right) \eta + \lambda_2 \frac{\partial f_2}{\partial \alpha_2} a_2 \right) dt \\
& + \left(-\lambda_1(\tau) + \lambda_2(\tau) + \frac{\partial \Psi_1}{\partial x} \right) \eta(\tau) + \left(-\lambda_2(T) + \frac{\partial \Psi_2}{\partial x} \right) \eta(T) \\
& + \left(\lambda_1(\tau)(f_1(x(\tau)) - f_2(x(\tau)) + \frac{\partial D}{\partial \tau} + \frac{\partial \Psi_1}{\partial x} f_2(x(\tau)) \right) \theta \\
& + \frac{\partial C_1}{\partial \alpha_1} a_1 + \frac{\partial C_2}{\partial \alpha_2} a_2.
\end{aligned}$$

This gives us the optimality conditions as

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

where the costates λ and ξ satisfy the following discontinuous (backwards) differential equations:

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

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-parametrization*, as given in the algorithm below:

```
Algorithm
Given $(t_0, T_1, r_1, \kappa_1), (T_1, T_2, r_2, \kappa_2), \ldots, (T_{p-1}, T_p, r_p, \kappa_p)$
Set $\tau_i(0) = T_i,\ i = 1, \ldots, p-1$
Initialize $\alpha_i(0),\ i = 1, \ldots, p$
Optimization $(k = 0)$
 Repeat
  Compute $x(t)$ forwards using $\alpha_i(k),\ \tau_i(k)$
  Compute $\lambda(t), \xi(t)$ backwards (including jumps)
  Compute $\frac{\partial J}{\partial \tau_i},\ \frac{\partial J}{\partial \alpha_i}$
  Gradient Descent
   Set $\tau_i(k+1) = \tau_i(k) - \gamma(k) \frac{\partial J}{\partial \tau_i}(\tau_i(k)),\ i = 1, \ldots, p-1$
   Set $\alpha_i(k+1) = \alpha_i(k) - \ell(k) \frac{\partial J}{\partial \alpha_i}(\alpha_i(k)),\ i = 1, \ldots, p$
  Set $k = k + 1$
 Until $\|\nabla J\| \leq \delta$
```

An example of this proposed approach is shown in Figure 3, in which an oscillator is switching between two modes - one slow (corresponding to walking) and one fast (corresponding to running). The frequencies and dynamics associated with the two modes are

$$\omega_1 = 5\frac{\alpha_1^2}{\alpha_1^2+10} + 5$$
$$\omega_1 = 10\frac{\alpha_2^2}{\alpha_2^2+10} + 10$$
$$\dot{x} = \begin{bmatrix} 0 & -\omega_{1,2} \\ \omega_{1,2} & 0 \end{bmatrix} x$$

and the cost is

$$J(\tau,\alpha_1,\alpha_2) = \int_0^3 0.1x(t)^T x(t)dt + (\tau-2)^2 + 0.1\alpha_1^2 + 0.2\alpha_2^2 + (x(\tau)-[-1,0])^T(x(\tau)-[-1,0]^T) + 5x(3)^T x(3).$$

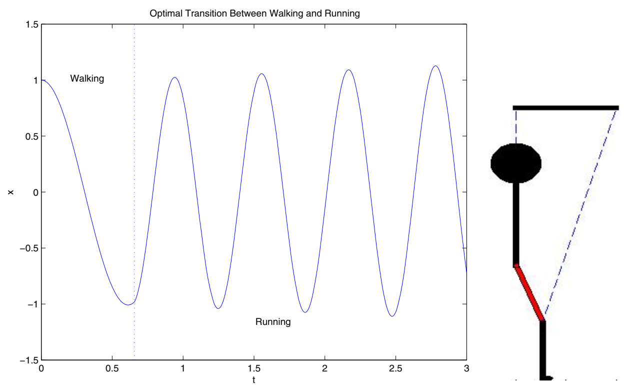


Fig. 3. A simplified locomotion model in which walking and running are defined through linear oscillators with different frequencies. The figure on the left depicts the waveform of the two gaits, and the figure on the right is a snapshot of the animation showing the result.

4 Motion and Caricature: *"Imitate, Simplify, Exaggerate"*

A system with limited expressive powers, such as a marionette, needs to be able to convey the proper emotions in such a way that a human audience understands what is being conveyed. As previously mentioned, puppeteers achieve this through the three steps: *Imitate*, *Simplify*, and *Exaggerate*. We will make these three steps formal in that human motion is being mimicked, after which the resulting motions are projected onto the constrained space over which the marionette operates (see Section 2.2), followed by a transformation of the resulting motion in such a way that the "energy" of the original (non-projected) motion is conserved.

4.1 Conservation of Energy

The method we propose for achieving this is by studying the way professional puppeteers actually control their puppets, as well as draw inspiration from human-like motions, since if the puppet is a human marionette, we would typically like to be able to execute human-like motions. However, since marionettes are constrained in such a way that they cannot be as expressive as human motions, we will first identify human motions (corresponding to the Imitate phase in puppetry) project human motion down onto the space of available puppet motions (the Simplify phase) and then exaggerate these motions in order to make them sufficiently expressive (the Exaggerate phase). Formally speaking, given a desired trajectory $z(t) \in Z$ that we would like the puppet (whose state is $x(t) \in X,\ \dim(X) \leq \dim(Z)$) to follow, we define a projection-like mapping $\rho : Z \rightarrow X$. The Simplify-phase thus consists of trying to minimize expressions like

$$\int_0^T L(x(t) - \rho(z(t)))dt,$$

subject to the puppet dynamics $\dot{x} = f(x, u)$, with u being the control input, and where T is the temporal duration of the movement.

Moreover, if TZ and TX are the tangent spaces associated with Z and X respectively, we define the energy conservation cost through the mapping $\phi : TX \rightarrow TZ$ in the following manner

$$\int_0^T \mathcal{E}(\phi(f(x(t), u(t)) - \dot{z}(t))dt,$$

and the combined Simplify-Exaggerate optimization problem becomes

$$\min_u \int_0^T (L(x - \rho(z)) + \mathcal{E}(\phi(f(x, u)) - \dot{z}))dt.$$

As an example, consider the situation when the puppet dynamics is given by the completely controllable linear control system

$$\dot{x} = Ax + Bu,\ x \in \mathbb{R}^n$$

with $z \in \mathbb{R}^m,\ m \geq n$. Moreover, let the projection ρ be given by a linear projection Pz and similarly let ϕ be given by a linear relation $E(Ax+Bu)$. The instantaneous cost thus becomes

$$\begin{aligned} L(x - \rho(z)) + \mathcal{E}(\phi(f(x, u)) - \dot{z}) &= \frac{1}{2}(x - Pz)^T Q(x - Pz) \\ &+ \frac{1}{2}(EAx + EBu - \dot{z})^T R(EAx + EBu - \dot{z}), \end{aligned}$$

given positive definite weight matrices Q, R.

Given free initial and final positions $x(0)$ and $x(T)$, this is a standard LQ-optimization problem that can be readily solved, and an example solution is shown in Figure 4. In that example,

$$\dot{z} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ -0.1 & -0.4 & -1 \end{bmatrix} z, \; z(0) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\dot{x} = \begin{bmatrix} 0 & -1.1 \\ 1.1 & -0.1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$Q = I_2, \; R = 0.01 I_3$$

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; E = \begin{bmatrix} 2/3 & 0 \\ 0 & 2/3 \\ 1/3 & 1/3 \end{bmatrix}$$

4.2 Conservation of "Emotive" Energy

In the previous discussion, *energy* was defined in terms of motion, but one can easily picture a somewhat more esoteric yet perhaps more relevant notion of *emotive energy*. In other words, the puppet is asked to capture a particular emotive state through its motion. But, due to its constrained configuration space, a direct mapping from human emotions to puppet emotions is not feasible. (Our faces, for example, have many degrees of freedom, while the puppets' have significantly less.) The same approach as previously discussed would still apply, but

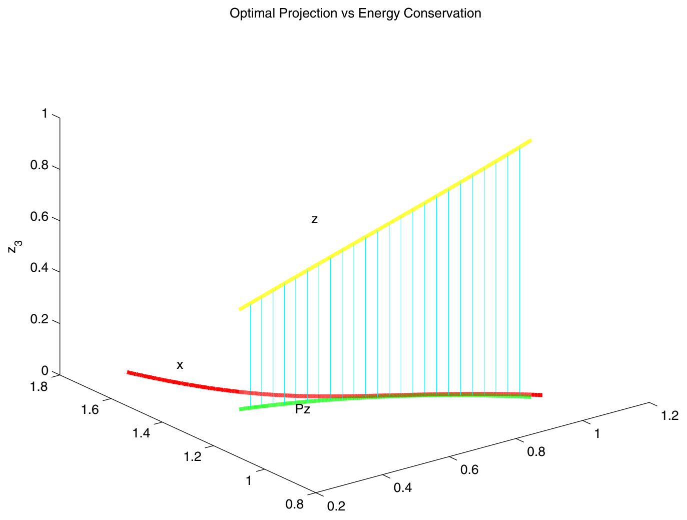


Fig. 4. This figure shows a simple example in which an optimal trade-off between tracking and energy-maintenance is achieved for linear systems. This method provides the basic building-block for the Simplify-Exaggerate phases of the construction of basic motion primitives.

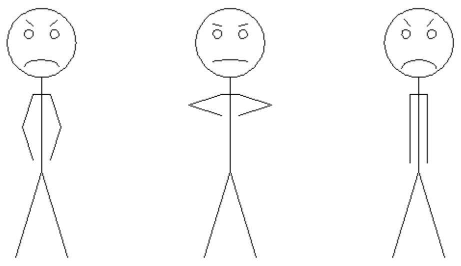


Fig. 5. *Anger* representation: The *Mathematica* stick-figure on the left depicts a nominal allocation between body and face movements

with the difference that now *emotive energy measures* will have to be generated. As a simple example, consider the stick-figure drawing in Figure 5. There the idealized puppet is asked to express *anger* and by constraining either the facial expressions or the body language, the *emotive energy* is maintained by either exaggerating the body or the facial movements. In the middle figure, the stick-figure has to work really hard to move its face, while in the right figure it has to work hard to move its arms. Because of this, the middle figure has a more mild facial expression but more dramatic body expression and the figure on the right has a severe facial expression with almost no body expression. The middle and right figures have been generated automatically (in *Mathematica*) from the left figure by an optimization process similar to what was discussed in the previous paragraphs.

5 Conclusions

In this paper we presented the motion description language MDLp, which is a MDL that allows for slightly more structured instructions, making it useful for specifying and encoding autonomous puppetry plays in a manner that is faithful to standard puppetry choreography. In particular, MDLp is a formal language whose strings, when parsed by a dynamical system, produces optimized, hybrid control laws corresponding to strings of motions, locations, and temporal durations for each agent. The paper is concerned with the development of this language as well as with an optimization engine for hybrid optimal control of MDLp strings, and with the generation of motion primitives within the "Imitate, Simplify, Exaggerate" puppetry paradigm.

References

1. R.C. Arkin. *Behavior Based Robotics.* The MIT Press, Cambridge, MA, 1998.
2. B. Baird. *The Art of the Puppet.* Mcmillan Company, New York, 1965.
3. A. Bicchi, A. Marigo, and B. Piccoli. Encoding Steering Control with Symbols. *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.

4. 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.
5. F. Bullo and A.D. Lewis. *Geometric Control of Mechanical Systems.* Number 49 in Texts in Applied Mathematics. Springer-Verlag, 2004.
6. C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, Norwell, MA, 1999.
7. Center for Puppetry Arts. http://www.puppet.org/.
8. M. Egerstedt. Motion Description Languages for Multi-Modal Control in Robotics. In *Control Problems in Robotics, Springer Tracts in Advanced Robotics* , (A. Bicchi, H. Cristensen and D. Prattichizzo Eds.), Springer-Verlag, pp. 75-90, Las Vegas, NV, Dec. 2002.
9. M. Egerstedt and C.F. Martin. Conflict Resolution for Autonomous Vehicles: A Case Study in Hierarchical Control Design. *International Journal of Hybrid Systems*, Vol. 2, No. 3, pp. 221-234, Sept. 2002.
10. 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.
11. L. Engler and C. Fijan. *Making Puppets Come Alive.* Taplinger Publishing Company, New York, 1973.
12. E. Frazzoli. Explicit Solutions for Optimal Maneuver-Based Motion Planning. *IEEE Conference on Decision and Control*, Maui, Hawaii, Dec. 2003.
13. E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries. *IEEE Transactions on Robotics*, Vol. 21, No. 6, pp. 1077–1091, Dec. 2005.
14. 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.
15. E. Johnson and T. Murphey. Dynamic Modeling and Motion Planning for Marionettes: Rigid Bodies Articulated by Massless Strings. Submitted to *ICRA*, 2007.
16. E. Klavins. A language for modeling and programming cooperative control systems. In *Proceedings of the International Conference on Robotics and Automation*, 2004.
17. M. Kloetzer and C. Belta. Hierarchical Abstractions for Robotic Swarms. *IEEE International Conference on Robotics and Automation*, Orlando, FL, 2006
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. 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.
21. G.J. Pappas, G. Laffierier, and S. Sastry. Hierarchically consistent control sytems. *IEEE Trans. Automatic Control*, 45(6):1144–1160, June 2000.
22. P. Tabuada and G. Pappas. Linear Time Logic Control of Discrete-Time Linear Systems. Accepted for publication in *IEEE Transactions on Automatic Control.*