To appear in ICAPS 2005
Workshop on Plan Execution

Safe Execution of Temporally Flexible Plans for Bipedal Walking Devices

Andreas Hofmann and Brian Williams

Computer Science and Artificial Intelligence Lab, MIT
32 Vassar St., rm. 32 - 275
Cambridge, MA 02139

hofma@csail.mit.edu, williams@mit.edu

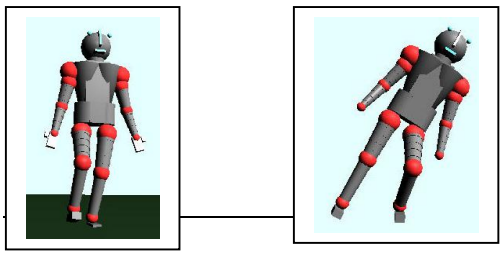
Abstract

Plans with temporal flexibility have been used to allow discrete systems to adapt to disturbances that occur while the plan is being executed. To control more complex devices, such as bipedal walking machines, we must extend this execution paradigm to the control of hybrid (discrete/continuous) systems. Systems of this type are difficult to control for two reasons: 1) their high dimensionality and nonlinearity make control complex, even under nominal circumstances; and 2) operation of such systems in unstructured environments requires robustness to significant disturbances.

We introduce a novel approach to hybrid temporally flexible plan execution that achieves robustness by transforming the high-dimensional system into a set of low-dimensional, weakly-coupled systems. This allows us to apply concepts previously developed for robust plan execution under uncertainty for discrete systems. We accomplish this decoupling using three components: 1) a feedback linearizing controller which provides basic decoupling of variables, 2) a hybrid plan dispatcher, which utilizes plan flexibility to adjust control settings for individual decoupled variables, and 3) plan compilation, which computes bounds for the dispatcher's adjustments to control settings that satisfy plan requirements. We show the interaction of these components in the control of a bipedal walking machine.

Introduction

Effective use of autonomous robots in unstructured, human environments requires that robots: 1) have sufficient autonomy to perform useful tasks independently, 2) have sufficient size, strength, and speed to accomplish such tasks in a timely manner, and 3) operate safely. A particularly challenging example of such a robot is a bipedal walking machine (Fig. 1a).



a. b.

Fig. 1 – a. Biped, b. reaction to disturbance

An example task for such a system is to walk to a soccer ball and kick it. If the system encounters a significant force disturbance, while performing this task, it will have to compensate by changing its stepping, or leaning the body, as shown in Fig. 1b. The disturbance may cause a delay, allowing another player to kick the ball, or it may interfere with movement synchronization; a lateral disturbance may cause synchronization problems with forward motion and stepping.

Movement is most naturally specified as a sequence of qualitative behaviors culminating in the task goal. For example, a walking gait cycle can be described as a sequence alternating between left single support (standing on the left foot), double support (standing on both feet), right single support, double support, etc. We call each step in such a sequence a *control epoch*, and the overall sequence, a *qualitative state plan*. Qualitative behavior within an epoch is specified with a set of state-space region and temporal range constraints. Thus, a qualitative state plan specifies a set of qualitatively similar trajectories, rather than a single trajectory through state-space.

Translating such a qualitative description into control actions that achieve the task goals is a challenging problem because the system is highly nonlinear, has high dimensionality and has input constraints that limit controllability. Additionally, the system must be robust to significant disturbances. Dynamic optimization techniques have been used to generate humanoid motion plans for animation applications [Popovic and Witkin, 1999]. However, these methods produce very detailed and inflexible reference trajectories, and are, therefore, not robust to disturbances. Robustness requires plan flexibility.

A powerful set of methods has been developed for discrete systems, for safe execution of temporally flexible plans [Morris, 2001]. These methods guarantee successful plan execution, as long as temporal uncertainty of activities is appropriately bounded. For example, suppose that the plan is for a car and sailboat to leave Boston for P-Town at the same time, and then to meet there. The duration of the sail is uncertain, but the uncertainty is bounded (6 to 12 hours). Likewise, the drive is 3 to 4

hours. Synchronization in P-Town is assured because the car can wait there indefinitely for the boat. In such systems, state is represented by logical variables. Constraints include logical constraints on these variables, and continuous temporal constraints. Executives for such systems operate by scheduling start times of activities. Key to this approach is the fact that activities can be started at any feasible time. This is rooted in the assumption that at the end of an activity (like driving), state will not change (the car will remain in P-Town).

This approach is appropriate for many kinds of applications, but it does not work for agile underactuated dynamic systems like bipeds, where movement is fast and controllability is limited. The lack of equilibrium points in such systems means that state is constantly changing, and the executive cannot assume the system will wait in a particular state at the end of an activity. For example, in dynamic walking, it is not possible to instantly stop forward movement in the middle of a step. The stepping foot must move out in front, or the biped will fall.

Systems of this type include continuous, as well as discrete state variables, so we refer to such systems as *hybrid*. Whereas an executive for a discrete system ignores the detailed dynamics of the system being controlled, an executive for a hybrid system must take these into account, along with associated controllability limits. Thus, in contrast to a discrete system executive, which controls the system by assigning start times to activities, a hybrid system executive controls timing indirectly, by adjusting control parameters for the dynamic system, and thereby, guiding trajectories to their goals at the right time.

Approach

The desired behavior is specified by a qualitative state plan. As with traditional temporal plans used for discrete systems [Muscettola et al., 1998], a qualitative state plan is composed of activities related by simple temporal constraints. However, a qualitative state plan differs in that an activity specifies the legal evolution of a state variable, rather than an executable activity [Bradley and Zhao, 1993]. This evolution is specified, qualitatively, in terms of state-space regions and temporal constraints. As shown in Fig. 2a, foot placement constraints define qualitative poses such as double support, or left single support, but the details of the joint positions and trajectories are omitted. A goal region for the forward position at the end of the gait sequence defines the goal. A temporal range constraint specifies task completion time requirements.

Such a qualitative behavior description is much more convenient than an activity plan, but translation into control actions is difficult. Executives for discrete systems solve this problem by calling a command library procedure associated with each activity. This is adequate for discrete systems, where dynamic performance is not of concern, but it is not for bipeds. We solve this problem by synthesizing a set of adaptive controllers, customized for the state plan. Performing such a synthesis on-line is not feasible because this process is computationally intensive. On the other hand, fixing all control parameters offline limits flexibility. Therefore, we use a mixed on-line/off-line approach, where an off-line plan compiler computes a set of bounds on control parameters, and an on-line hybrid dispatcher efficiently adapts control settings, within these bounds, to respond to disturbances.

To simplify controller synthesis, we utilize a feedback linearizing multivariable controller, which transforms the highly nonlinear, tightly coupled biped system into a loosely coupled set of linear 2nd-order single-input single-output (SISO) systems [Hofmann, 2004]. This *SISO abstraction* greatly simplifies the plan compiler's job, because the control parameters computed are for a small set of linear control laws, rather than for a complex nonlinear system.

The plan compiler and hybrid dispatcher comprise a model-based executive [Williams and Nayak, 1997], as shown in Fig. 2b. The next section describes the state plan and plant in more detail. We follow this with detailed descriptions of the plan compiler and hybrid dispatcher algorithms. We conclude with a presentation of results and discussion.

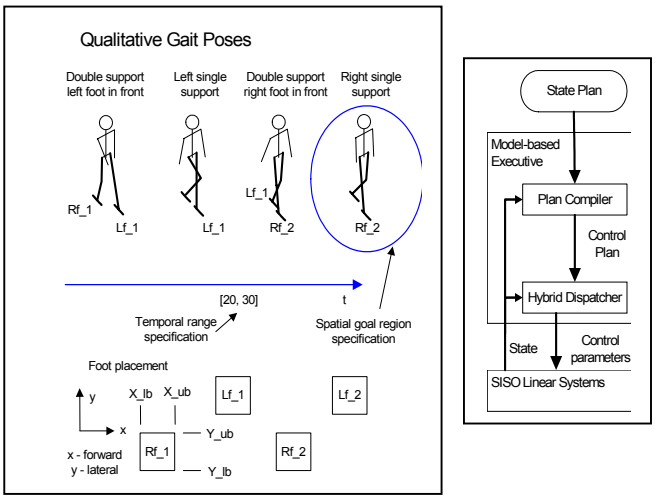


Fig. 2 – a. Qualitative task specification (left), b. Model-based executive (right)

Qualitative State Plan and Plant

A qualitative state plan is used to represent the desired state evolution of the plant (the system being controlled). Given a state plan and a plant, the execution task is to generate a sequence of control actions that moves the plant to a state consistent with that required by the plan.

Plant

The plant is represented by a set of dynamic equations that specify state evolution as a function of inputs. We define a

plant by the tuple $\langle \mathbf{x,y,u,h,f} \rangle$ where $\mathbf{x,y,u}$ is the set of state, output (controlled), and input variables, respectively, $\mathbf{h}$ is a set of algebraic equations that relate outputs to inputs and state, and $\mathbf{f}$ is a set of 1st-order differential equations that describe state evolution. The input vector, $\mathbf{u}$, includes control inputs, such as joint torques, and also environment inputs (ground contact forces). The state vector, $\mathbf{x}$, includes continuous variables, such as joint positions, and discrete variables, to indicate the presence of environment forces. The output vector, $\mathbf{y}$, contains the variables to be controlled, such as forward and lateral center of mass (CM) position. Our simulated walking biped has 18 degrees of freedom, and is highly nonlinear [Hofmann et al., 2002].

Qualitative State Plan

A qualitative state plan specifies state evolution using sequences of *activities* as shown in Fig. 3. Each row in this diagram specifies behavior for a particular quantity. Two types of quantities, *controlled quantities* and *input quantities*, can be specified. Controlled quantities are position/velocity pairs, corresponding to elements of $\mathbf{y}$. Input quantities are scalar functions, $g_i(y_i, \dot{y}_i)$, that represent control laws for controlled quantities, and, thus, correspond to elements of $\mathbf{u}$. In Fig. 3, forward and lateral CM are examples of controlled quantities, and forward and lateral CP (center of pressure) are examples of input quantities.

Each activity is part of a control epoch (column in Fig. 3). The control epochs in Fig. 3 correspond to the qualitative poses shown in Fig. 2. Thus, epoch 1 represents double support with the left foot in front, epoch 2, left single support, epoch 3, double support with the right foot in front, and epoch 4, right single support. Epoch 5 repeats epoch 1, but is one gait cycle forward. Vertical bars in Fig. 3, between rows, represent synchronization constraints, such that all quantities advance to the next control epoch at the same time.

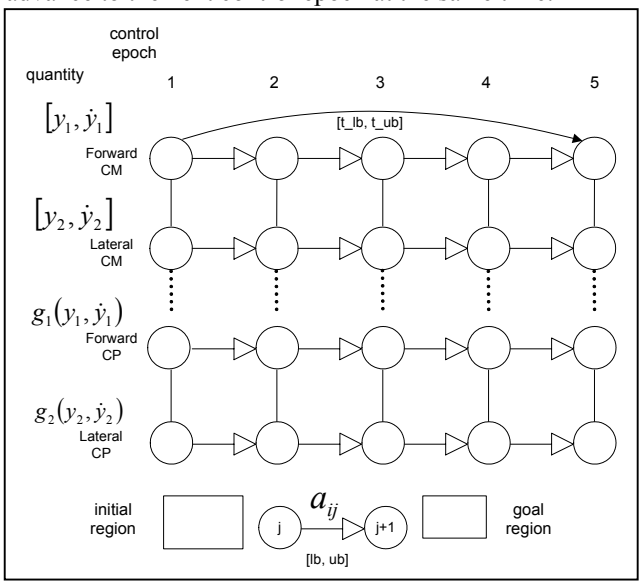


Fig. 3 – Qualitative State Plan

Each activity may have a temporal duration range constraint, indicated by [lb, ub] (lb indicates lower bound, ub, upper bound). In addition, range constraints initial and goal regions for quantities may be specified. For controlled quantities, regions are specified using rectangles in position/velocity phase space. For input quantities, regions are specified using scalar ranges. Note that the duration and region constraints are optional. In fact, these constraints are omitted for many activities. In Fig. 3, we care about the initial and final region of the CM, but not about the details in between. Similarly, we care that the gait cycle be completed within time range [t_lb, t_ub], but not about the specific duration of each activity.

Each activity may also have a *tube* constraint specifying a required region for the associated quantity over the entire duration of the activity. Such tube constraints are useful, for example, for limiting the range of forward and lateral CP. This is important because CP is an input quantity that is limited by foot placement (see also Fig. 2a). For example, in epoch 2 (left single support), the CP is restricted to the support region under the left foot.

Formally, we define a state plan as a set, A, of activities, $a(i,j)$, where i refers to the quantity, and j to the control epoch. An activity is defined by the tuple $\langle R_{init}, R_{tube}, R_{goal}, R_{temporal}, a_{next} \rangle$ where R_{init}, R_{tube}, and R_{goal} specify, respectively, initial, operating, and goal regions in state space for the controlled variable associated with the activity, and $R_{temporal}$ specifies temporal constraints. The activity a_{next} is the activity to transition to when the current activity is finished.

The region constraints, R_{init}, R_{tube}, and R_{goal} are of the form $(y_{i\min} \le y_i \le y_{i\max}) \wedge (\dot{y}_{i\min} \le \dot{y}_i \le \dot{y}_{i\max})$ for controlled quantities, and $(y_{i\min} \le y_i \le y_{i\max})$ for input quantities. Use of such unary constraints with constant bounds results in rectangular regions in position/velocity state-space. This implies more conservative bounds than would be possible if the constraints were multivariable and nonlinear, but it also greatly simplifies planning. An activity may begin if its quantity is in the region defined by R_{init}. An activity cannot end unless the quantity is in R_{goal}. The quantity must stay within R_{tube} for the entire duration of the activity.

$R_{temporal}$ is of the form $\langle R_{duration}, A_{parallel} \rangle$, where $R_{duration}$ is a simple temporal constraint $[lb, ub]$, and $A_{parallel}$ is a set of activities that must finish simultaneously with the current one (vertical bars in Fig. 3). For example, for a biped, movement of the stepping foot must be synchronized with forward movement of the center of mass.

A valid plan execution is defined as follows. An activity finishes if its R_{goal}, $R_{duration}$, and $A_{parallel}$ constraints are satisfied. After it finishes, it transitions to the successor, a_{next}, immediately. An activity, a, is executed successfully iff there exists a start time, ts, and a finish time, tf, for the activity, such that $lb \le tf - ts \le ub$, and there exists a trajectory for the associated controlled variable y such that $y(ts)$ and $\dot{y}(ts)$ satisfy R_{init}, $y(tf)$ and $\dot{y}(tf)$ satisfy R_{goal}, and $y(t)$ and $\dot{y}(t)$ satisfy R_{tube}

for $ts \le t \le tf$. A state plan is executed successfully iff each activity, $a(i,j)$, is executed successfully, and the associated finish time, $tf(i,j)$, is such that if the activity has a successor, $a(i,j+1)$, then $tf(i,j) = ts(i,j+1)$, and for any parallel activity, $a(k,j)$, listed in $A_{parallel}$, $tf(i,j) = tf(k,j)$.

Plan Compiler

A feasible state trajectory for a state plan is produced by adjusting control parameters for each SISO controller associated with one of the controlled quantities in the state plan. To improve run-time performance, restrictions on these control parameters are determined at compile time and are captured in a control plan, where the activities of a control plan are in a one-to-one correspondence with those in the state plan. The precise control parameter setting for each control plan is then determined by the dispatcher at run-time.

A qualitative control plan contains all the information needed to control the SISO system, and to monitor its status with respect to region and temporal bounds. A control plan consists of the activities from the state plan, with two additions: 1) control information is included, and 2) the region constraints, R_{init}, R_{tube}, and R_{goal}, and the duration constraint, $R_{duration}$, specify non-infinite bounds. Control parameter constraints are of the form $\langle k_{1\min}, k_{1\max}, k_{2\min}, k_{2\max} \rangle$, and specify bounds on control parameters for the linear control laws used in the SISO abstraction.

In computing bounds on the control parameters, the compiler performs an adaptive controller synthesis. The hybrid dispatcher utilizes the flexibility of the control parameter ranges to adapt control settings as needed. In order to maximize robustness to disturbances, the compiler attempts to maximize the size of initial regions, and tubes, minimize the size of goal regions, and maximize controllable temporal activity duration ranges. Maximizing initial regions and minimizing goal regions results in a contraction; the family of trajectories in the tube "contract" to each other as time advances, a property that provides stable recovery after a disturbance. Maximizing controllable temporal range makes synchronization with other controlled quantities easier.

In generating trajectories, the compiler must take into account dynamics. We want fast performance, but due to the underactuated nature of the system, this leads to a reduction in controllability. Thus, future consequences of current actions are important, and the compiler must take into account future epochs in its computation. The compiler proceeds in two steps. First, it computes a nominal trajectory that reaches the goal state from the initial state, and that satisfies all constraints. Second, to provide robustness to disturbances, the compiler expands the nominal trajectory, creating regions and tubes not specified explicitly in the qualitative state plan, attempting to maximize initial regions, minimize goal regions, and maximize controllable temporal range. For both steps, we utilize an SQP (Sequential Quadratic Programming) optimizer, and formulate the problem as an NLP (Nonlinear Program). These two steps are now described in detail.

Nominal Trajectory Computation

The NLP formulation for the nominal trajectory computation (first step of compiler) is as follows. For each activity, a_{ij}, associated with a controlled quantity, parameters to optimize are $\langle y_{init}, \dot{y}_{init}, y_{goal}, \dot{y}_{goal}, t_{goal}, k_1, k_2 \rangle$, where $y_{init}, \dot{y}_{init}$ are the initial state of the quantity associated with the activity, $y_{goal}, \dot{y}_{goal}$ are the final state, t_{goal} is the duration, and k_1, k_2 are parameters for the linear control law. Constraints are as follows. The trajectory is defined by the analytic solution to the linear 2nd-order differential equation formed by applying the linear control law to the SISO abstraction. This yields an equality constraint that relates goal to initial state:

$$\begin{aligned} y_{goal} &= f_1(y_{init}, \dot{y}_{init}, k_1, k_2, t_{goal}) \\ \dot{y}_{goal} &= f_2(y_{init}, \dot{y}_{init}, k_1, k_2, t_{goal}) \end{aligned} \quad (1)$$

Continuity from one epoch to the next is expressed as

$$\begin{aligned} y_{goal}(i,j) &= y_{init}(i,j+1) \\ \dot{y}_{goal}(i,j) &= \dot{y}_{init}(i,j+1) \end{aligned} \quad (2)$$

Inequality constraints for R_{init}, R_{tube}, R_{goal}, and $R_{duration}$ are as described previously. Synchronization constraints across quantities are expressed as

$$\forall i,j : t_{goal}(i,j) = t_{goal}(i+1,j) \quad (3)$$

Finally, the temporal constraint on overall state plan execution time is given by

$$\forall i,j : t_{lb} \le \sum_j t_{goal}(i,j) \le t_{ub} \quad (4)$$

The cost function includes terms that maximize the distance to the R_{tube} boundaries.

Qualitative Control Plan Computation

The NLP formulation for the control plan computation (second step of compiler) is as follows. For each activity, a_{ij}, parameters to optimize are: $\langle \dot{y}_{init\min}, \dot{y}_{init\min}, y_{init\max}, \dot{y}_{init\max} \rangle$ (parameters of R_{init}), $\langle y_{goal\min}, y_{goal\min}, y_{goal\max}, y_{goal\max} \rangle$ (parameters of R_{goal}), $\langle t_{\min}, t_{\max} \rangle$ (parameters of $R_{duration}$), and $\langle k_{1\min}, k_{1\max}, k_{2\min}, k_{2\max} \rangle$, the bounds on the control parameters. In order to understand how this computation works, it is necessary to understand two trajectories that represent extremes of behavior: the *guaranteed fastest trajectory* (GFT), and the *guaranteed slowest trajectory* (GST). The GFT represents a lower bound on the time needed to get from anywhere in the initial region, to somewhere in the goal region. The GST represents the corresponding upper bound. For both these trajectories, it is assumed that velocity does not change sign.

Consider the initial and goal regions shown in Fig. 4a. For the GFT, the worst-case starting point in the initial region is point B, which corresponds to $y_{init\min}, \dot{y}_{init\min}$. This represents the slowest possible start. By accelerating as quickly as possible, the GFT reaches point D, which corresponds to $y_{goal\min}, \dot{y}_{goal\max}$. This represents the

fastest finish point in the goal region. For the GST, the worst-case starting point is point A, which corresponds to $y_{init\max}, \dot{y}_{init\max}$. This represents the fastest possible start. By accelerating as slowly as possible, the GST reaches p;oint C, which corresponds to $y_{goal\max}, \dot{y}_{goal\min}$. This represents the slowest finish point.

The times for each trajectory are designated t_{GFT} and t_{GST}. If $t_{GFT} < t_{GST}$, then there exists a temporal range, $[t_{GFT}, t_{GST}]$, during which the endpoint of a trajectory beginning from anywhere in the initial region can be guaranteed to be in the goal region. The existence of this temporal range is important for synchronizing with other controlled quantities. Thus, the GFT and GST are useful for determining a maximum initial region, given a particular goal region, such that the controllable temporal range exists.

GFT and GST can be understood intuitively by considering a very simple control law. Suppose that the only control input allowed is a single acceleration spike. If this spike is applied at the beginning of the trajectory, then maximum velocity is reached immediately, resulting in the GFT, as shown in Fig. 4b. If this spike is applied at the end, then the trajectory will progress at minimum velocity resulting in the GST.

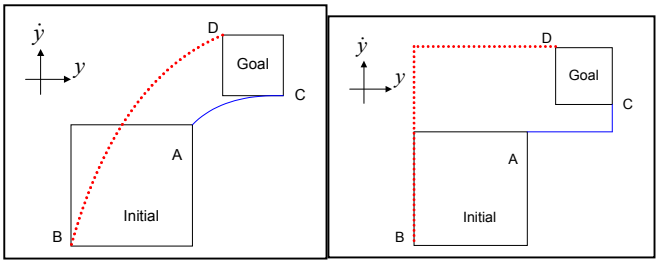


Fig. 4 – a. GFT (dotted) and GST (solid), left, b. for acceleration spike control laws, right

In the NLP formulation, existence of initial and goal regions is expressed as

$$y_{init\min} < y_{init\max}, \quad \dot{y}_{init\min} < \dot{y}_{init\max} \qquad (5)$$
$$y_{goal\min} < y_{goal\max}, \quad \dot{y}_{goal\min} < \dot{y}_{goal\max}$$

To guarantee contraction, the goal region of an activity must fit inside the initial region of its successor.

$$\begin{aligned} y_{goal\min}(i,j) &\geq y_{init\min}(i,j+1) \qquad (6)\\ \dot{y}_{goal\min}(i,j) &\geq \dot{y}_{init\min}(i,j+1)\\ y_{goal\max}(i,j) &\leq y_{init\max}(i,j+1)\\ \dot{y}_{goal\max}(i,j) &\leq \dot{y}_{init\max}(i,j+1) \end{aligned}$$

Constraints representing the GFT are expressed as

$$\begin{aligned} y_{goal\min} &= f_1(y_{init\min}, \dot{y}_{init\min}, k_{1\max}, k_{2\max}, t_{\min}) \qquad (7)\\ \dot{y}_{goal\max} &= f_2(y_{init\min}, \dot{y}_{init\min}, k_{1\max}, k_{2\max}, t_{\min}) \end{aligned}$$

Constraints representing the GST are expressed as

$$\begin{aligned} y_{goal\max} &= f_1(y_{init\max}, \dot{y}_{init\max}, k_{1\min}, k_{2\min}, t_{\max}) \qquad (8)\\ \dot{y}_{goal\max} &= f_2(y_{init\max}, \dot{y}_{init\max}, k_{1\min}, k_{2\min}, t_{\max}) \end{aligned}$$

The requirement for temporal controllability is expressed as $t_{\min} < t_{\max}$. Synchronization constraints are

$$\begin{aligned} &(t_{\min}(1,j) \leq t_{trans\min}(j) \leq t_{trans\max}(j) \leq t_{\max}(1,j)) \wedge \ldots\\ &(t_{\min}(i,j) \leq t_{trans\min}(j) \leq t_{trans\max}(j) \leq t_{\max}(i,j)) \end{aligned}$$

Thus, $[t_{trans\min}(j), t_{trans\max}(j)]$ is the temporal range when transition out of control epoch j may occur.

The cost function maximizes initial region size, minimizes goal region size, and maximizes controllable temporal range.

Hybrid Dispatcher

The hybrid dispatcher executes the qualitative control plan. It acts as an adaptive controller, adjusting control parameters within the limits specified in the control plan, to guide each controlled quantity into its goal region at the correct time. When all quantities are in their respective goal regions, the hybrid dispatcher transitions to the next control epoch. Note that unlike dispatchers for discrete systems [Morris, 2001], the hybrid dispatcher is not able to directly schedule start times for activities. Rather, it indirectly controls timing by adjusting control parameters.

At the beginning of a new control epoch, j, the dispatcher computes a target transition time, $t_{trans}(j)$. When there is no temporal uncertainty, it chooses a time in the range $[t_{trans\min}(j), t_{trans\max}(j)]$. When there is uncertainty in one of the controlled quantities, the transition time is determined when the uncertainty for this controlled quantity is resolved.

For each controlled quantity, the dispatcher then monitors progress by computing a prediction of the point in state space for the controlled quantity at $t_{trans}(j)$. This prediction is computed analytically in the same manner as eq. 1, so it is fast. If the predicted point is within the goal region, then the dispatcher does nothing. If it is outside the goal region, then the dispatcher adjusts the control parameters to attempt to move the predicted point back into the goal region. If this is unsuccessful, the plan execution fails, and the dispatcher requests a new plan. If all trajectories execute successfully, then when all controlled quantities are in their respective goal regions, the dispatcher transitions to the next control epoch.

Results and Discussion

Fig. 5a shows a nominal trajectory for CM and CP, computed by the first step of the plan compiler. The vertical axis points to the model's left, and the horizontal axis points forward. The lateral CP trajectory first rises, during double support, and reaches its maximum value at left single support. It then falls during the subsequent double support control epoch as weight is shifted to the right foot.

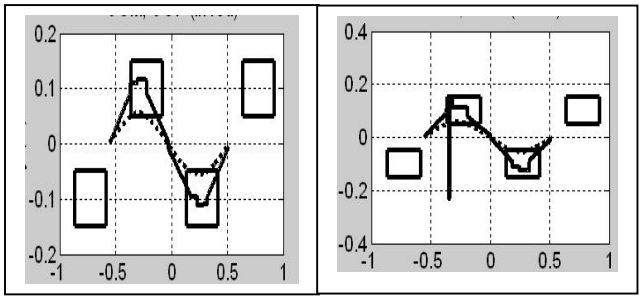


Fig. 5 – a. Lateral vs. forward CP (solid) and CM (dotted) nominal trajectories, left, b. trajectories for large disturbance, right, (foot position shown by rectangles)

Fig. 6 shows control plan initial and goal regions computed by the second step of the plan compiler for control epochs 1-4. Fig. 6a shows lateral CM (velocity vs. position), and Fig. 6b shows forward CM.

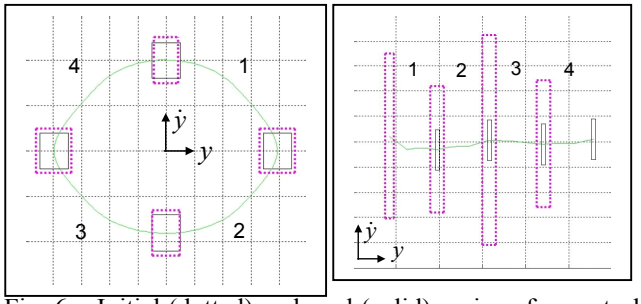


Fig. 6 – Initial (dotted) and goal (solid) regions for control epochs 1 – 4, a. lateral CM (left), b. forward CM (right), plots show region in velocity-position phase space.

As can be seen in Fig. 6b, good contraction is achieved for forward CM; the goal regions fit well inside the initial regions for the subsequent mode. Contraction is not as good for lateral CM; the goal regions barely fit inside the initial regions (Fig. 6a). This is due to the fact that controllability in the lateral direction is more limited, because the support base is narrower (in single support, the foot is longer than it is wide).

Temporal Uncertainty

If controllability is very limited, as in Fig. 6a, it may be difficult to achieve a large enough initial region to ensure contraction. This situation can be improved by relaxing the constraint $t_{GFT} < t_{GST}$, as shown in Fig. 7.

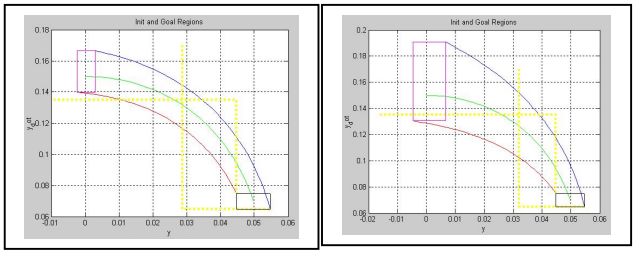


Fig. 7 – a. t(GFT) = t(GST), left, b. t(GFT) > t(GST), right, resulting in larger initial region

The intuitive explanation for why this happens is that as GFT is allowed to take longer, the B point of the initial region stretches to the left (position decreases). Similarly, if GST is allowed to complete faster, point A moves to the right.

Unfortunately, relaxing this constraint means that we lose temporal controllability; the time of arrival in the goal region can no longer be precisely controlled, making synchronization more difficult. However, the uncertainty on arrival time is bounded by $[t_{GST}, t_{GFT}]$. This may still be okay if controllability of other controlled quantities is strong enough to compensate. When this is the case, the system is dynamically controllable [Morris, 2001]. This is represented using an STNU (simple temporal network with uncertainty), as shown in Fig. 9.

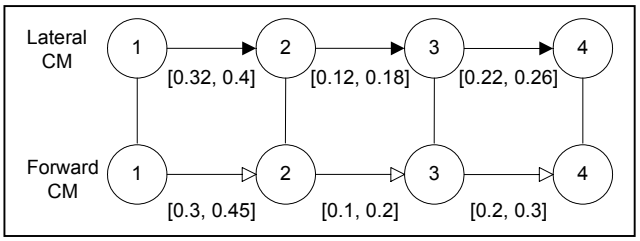


Fig. 9 – STNU for lateral and forward CM

In this STNU, the temporal durations for lateral CM are uncertain but bounded. The temporal durations for forward CM are certain, and they are large enough to compensate for the uncertainty in the lateral CM. Thus, transition synchronization can still be guaranteed, though this will require adjustment by the hybrid dispatcher once the uncertainty is resolved.

Our tests on the simulated biped show that a small lateral CM disturbance (100 N for 0.01 sec) can be handled by the multivariable controller without any immediate action by the dispatcher. A larger (250 N) disturbance requires gain adjustment by the dispatcher. A disturbance of 300 N, shown in Fig. 5b, is too large for the dispatcher to handle by itself; a new dispatchable plan is required. Additional tests show that the biped can walk at 0.5 m/s on ground with largely varying degrees of softness.

These tests demonstrate compliance to disturbances at three levels: 1) use of low-gain control, 2) flexibility in the dispatchable state plan allowing for adjustments by the hybrid dispatcher, and 3) fast reactive planning. This allows for integrated handling of disturbances of varying degrees of severity.

References

[Bradley and Zhao, 1993] E. Bradley and F. Zhao. Phase-space control system design. *Control Systems*, 13(2),39-46 April, 1993

[Goswami, 1999] A. Goswami. Postural stability of biped robots and the foot rotation indicator (FRI) point. *International Journal of Robotics Research*, July/August 1999

[Hofmann et al., 2002] A. Hofmann, M. Popovic, H. Herr. Humanoid Standing Control: Learning from Human Demonstration. *Journal of Automatic Control*, 12(1), 16–22

[Hofmann et al., 2004] A. Hofmann, S. Massaquoi, M. Popovic, and H. Herr. A sliding controller for bipedal balancing using integrated movement of contact and non-contact limbs. *Proc. International Conference on Intelligent Robots and Systems* (IROS). Sendai, Japan

[Morris et al., 2001] P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal

uncertainty. *Proceedings of the 17th International Joint Conference on A.I.* (IJCAI-01). Seattle (WA, USA).

[Muscettola et al., 1998] N. Muscettola, P. Morris, and I. Tsamardinos. Reformulating temporal plans for efficient execution. *Proc. Of Sixth Int. Conf. On Principles of Knowledge Representation and Reasoning*, 1998

[Popovic et al., 1999] Z. Popovic and A. Witkin. Physically based motion transformation. Siggraph 1999

[Williams and Nayak, 1997] B. Williams and P. Nayak. A Reactive Planner for a Model-based Executive. *Proceedings of the International Joint Conference on Artificial Intelligence* (IJCAI, 1997)