Combining Automated Planning and Hybrid Control: A Quadruped Bouncing Gait

Robert Effinger

Model-based and Embedded Robotic Systems Group
Computer Science and Artificial Intelligence Lab, MIT
effinger@mit.edu

Abstract

This proposed thesis topic aims to combine automated planning and hybrid control techniques. In this paper we describe a particular case study in this direction: the development of a quadruped bouncing gait with four qualitative states, quad-stance, touch-off, flight, and touch-down. Our approach is novel in that gait achievement is defined in terms of flexible constraint windows in state-space and time. This key feature enables the use of a controller that chooses from entire sets of optimal joint motion trajectories at execution time instead of tracking a single pre-planned optimal trajectory with high-impedance. In simulation, we demonstrate the robustness of this approach by maintaining a steady-state quadruped bouncing gait despite significant disturbances at execution time. The focus of this thesis work will be to investigate techniques to interleave and concurrently execute such plans in order to achieve complex and abstract goal specifications.

Introduction

This paper features the development of a sagittal plane model of a bouncing quadruped with actuation limits and four qualitative states, quad-stance, touch-off, flight, and touch-down, as depicted in Figure 1. A control policy for the template was developed in two phases. First, the bouncing gait was defined in terms of its qualitative states, called a *Qualitative State Plan (QSP)* [3]. Transitions between the qualitative states in a QSP occur through flexible constraint windows in state-space and time. For example, the center of mass of the quadruped, the red dot in Figure 1, passes through the blue constraint windows at appropriate times in order to maintain steady-state bouncing. Second, a controller was developed for each qualitative state that ensures the quadruped passes through the constraint windows under bounded disturbances. Steady-state bouncing was achieved in simulation.

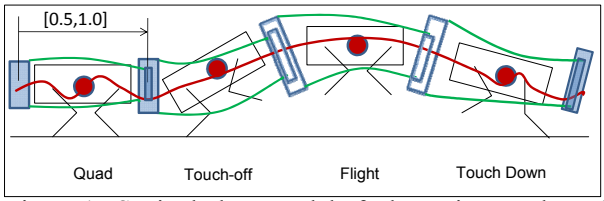

Figure 1: Sagittal plane model of a bouncing quadruped with four qualitative states.

In the Future Work section of this paper, we hypothesize ways to extend this approach to be applicable to multiple cooperating agents that can achieve complex and abstract goal specifications.

Prior Work

This thesis proposal builds upon prior work by Andreas Hofmann and Brian Williams of the Model-based and Embedded and Robotic Systems group at MIT [1,2,3]. The overarching objective of these cited works is to combine automated planning and hybrid control techniques. For example, this previous work has considered problems such as getting a walking biped to kick a soccer ball and to recover from trip disturbances. In this study, we employ these same techniques to a new problem; demonstrating a steady-state bouncing quadruped gait. In the next two sections, we describe why QSPs provide an intuitive link between automated planning and hybrid control.

Automated Planning with QSPs

Automated planning techniques traditionally assume a library of activities or actions (in our case QSPs). The automated planner then generatively constructs a plan, from among the library of alternatives, by piecing together pre and post-conditions. As depicted in Figure 2, for example, to get from the start to the goal, a quadruped may choose to walk forward, turn right, and then bounce to the goal, by piecing together the goal and initialization regions of successive QSPs. There are often many possible alternatives to achieve the goal given a library of actions.

The appeal of approach is in reducing the search complexity from a large continuous problem over the entire state-space (with a branching-factor of all possible

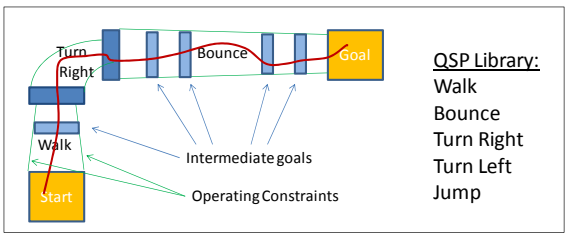

Figure 2: Automated planning given a library of QSPs.

control actions at each time step), to a parameterized discrete search over a library of possible actions with finite but flexible temporal durations. Also note that the solution of this approach does not prescribe a single optimal reference trajectory, but instead outlines a set (or bundle) of trajectories with flexible constraint windows in state-space and time. We think this approach is significant because it:

1.) Generates plans that ensure the inherent hybrid constraints of underactuated robots are obeyed.
2.) Provides a computationally tractable way to generate and then compare plans with shared objectives, interactions, and concurrent threads of activities.

Note that this approach outputs only valid interconnections of trajectory primitives, and is similar in concept to the Maneuver Automaton by Frazzoli [4]. In the Future Work section, we hypothesize how extending this approach to multiple coordinating agents is similar in concept to planning over and verifying the correct execution of multiple, concurrently executing Maneuver Automatons.

Hybrid Control with QSPs

In this section, we describe our hybrid control approach in more detail. We call a control policy that executes a Qualitative State Plan, a *Qualitative Control Plan (QCP)*. As depicted in Fig. 3, a QCP is a piecewise linear mapping from robot state-space (only those states within the QSP) to control inputs that guarantee achievement of the goal region (under bounded disturbance). In this project, we use a technique called multi-parametric optimization in order to develop the controllers that comprise the QCP. Multi-parametric optimization employs a Linear Quadratic Regulator extended to incorporate linear inequality constraints [5] in order to develop cost-optimal solutions for systems of low complexity. Implementation-wise, we used a freely available and user-friendly Matlab toolbox, called the Multi-parametric toolbox (MPT) [6], which was developed by the Automatic Control Laboratory at the Swiss Federal Institute of Technology (ETH).

h This approach enables us to find optimal control policies for simple, low-dimensional systems, given linear constraints on initial, goal, and operating regions. For higher dimensional and nonlinear templates, it is also possible to use techniques such as dynamic programming, value iteration, and policy search. Note that as long as the trajectory remains within the QCP, plan success and dynamic stability are guaranteed.

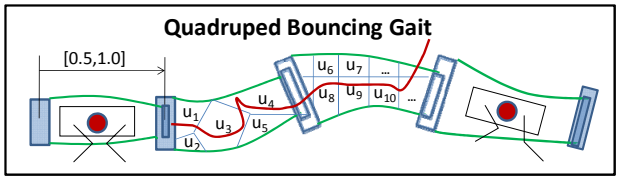

Figure 3: A QCP partitions the QSP's state-space into regions, and defines a control action for each region.

To give an example, we depict in Fig. 4 below, a 3D projection of the 6D controller $(x, \dot{x}, y, \dot{y}, \theta, \dot{\theta})$ for the flight to touch-down phase of the quadruped bouncing gait. X_1 is "x", X_2 is "y" and X_3 is "θ". In addition, two trajectories (starting from different initial conditions) are superimposed and shown with red and blue lines in Fig. 4. These trajectories depict how control actions are chosen at each time-step (green dots) to guide trajectories optimally (from different initial conditions) towards the goal. Notice that X_2 (the height of the quadruped's COM) is only defined from 4 to 0.4 units, respecting our linear constraint that its COM remain greater than 0.4 and less than 4 units.

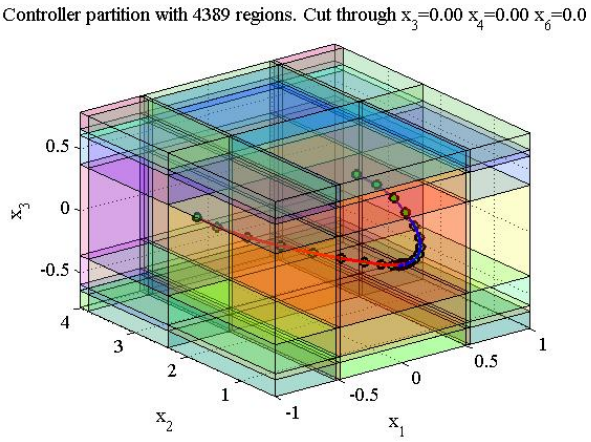

Figure 4: Flight to Touch-down COM Controller with two superimposed trajectories.

Results

A steady-state 2D sagittal-plane quadruped bouncing gait was achieved in simulation. Our work towards this result is summarized in four steps.

1. Develop a simple 2D quadruped simulation.
2. Develop a vertical bouncing gait.
3. Develop a forward moving bouncing gait.
4. Add noise to the gaits to show robustness.

Figure 5 provides a glimpse of the quadruped simulation and results. We forego a more detailed explanation in order to discuss the most important aspect of this paper, potential directions for future work and collaborations.

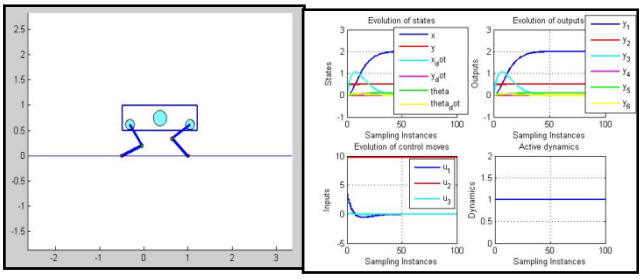

Figure 5: Quadruped simulation and MPT analysis.

Future Work and Potential Collaborations

In this section, we outline the research directions that we wish to pursue, provide comparisons to existing work that we are aware of, and invite researchers to inform us of other related work that we are not yet aware of. The overarching objective of this research can be stated as follows:

> "To develop an autonomous model-based executive that coordinates multiple autonomous agents in order to achieve complex and abstract goal specifications, while responding robustly to disturbances and failures at execution time."

This broad problem statement cuts across much of current AI and automated planning research. More specifically, there are several aspects of the problem that we would like to focus on, in-particular:

- Extending our current model-based approach to handle complex and abstract goal specifications involving multiple agents.
- Planning autonomously for agents that are *high-dimensional, underactuated* and *hybrid* in nature.
- Developing a model-based executive that responds robustly to disturbances and failures at execution time, in order to maintain progress towards achieving the goals.

To accomplish these objectives, we propose a model-based executive which takes as inputs a goal specification and plant models for each agent. The model-based executive then reasons from the models to generate and execute a plan that accomplishes the goal specification.

Informally, we identify and discuss the three key components of this architecture: 1.) the goal specification, 2.) the plant model, and 3.) the model-based executive.

1.) The Goal Specification

The goal specification we propose needs to be capable of supporting complex and abstract goals. In addition, it needs to support flexible state-space and timing constraints. Consider one such example: "Robot Alpha should visit either region 2 or region 3, and then meet up with Robot Beta at region 4 within 5 minutes. Then they should proceed to the closest recharging station, and recharge for at least 10 minutes but no more than 20 minutes each."

In the literature, there are many types of specification languages, motion description languages [7], temporal logics [8], and reactive programming languages [9]. At present, our research group uses the Reactive Model-based Programming Language [10], an in-house language which shares in many aspects of the above languages.

2.) The Plant Model

Fundamentally, the only requirement we impose on the plant model, is that it can model a hybrid, and potentially high-dimensional and underactuated, system. This plant model could consist simply of a library of primitive QSPs, and an algorithm that is capable of piecing them together. Or alternatively, we could pre-compile offline all valid interconnections of trajectory primitives into an automaton, similar to the Maneuver Automaton approach [4]. We could also consider using model-checking formalisms such as the Timed Abstract State Machine language [11], or perhaps Markov Decision Processes [12].

3.) The Model-based Executive

The fundamental requirement for our model-based executive is the ability to generate and execute plans that achieve the goal specification, despite disturbances and failures at execution time. Key to meeting this requirement will be the ability to reason over multiple plant models that are operating concurrently in order to generate a plan that achieves the goal specification, while simultaneously obeying the dynamical and environmental constraints inherent to each plant.

The model-checking, verification, and AI communities have done a lot of research on concurrently running and hierarchical state machines that may be applicable [13,14]. It is still unclear to us how much prior work might be leveraged in developing our model-based executive, and what, if any, are the limitations of prior approaches. This is an important topic for us to address as we move forward.

References

[1] A. Hofmann, S. Massaquoi, M. Popovic, and H. Herr. A Sliding Controller for Bipedal Balancing Using Integrated Movement of Contact and Non-Contact Limbs. In *IROS 2004*, Sendai, Japan, Oct. 2004.

[2] A. Hofmann, and B. Williams. Exploiting Spatial and Temporal Flexibility for Plan Execution of Hybrid, Under-actuated Systems. In *AAAI 2006*, Boston, MA, USA, July 2006.

[3] A. Hofmann Robust Execution of Bipedal Walking Tasks from Biomechanical Principles In *Ph.D. Thesis*, MIT, Dec. 2005.

[4] E. Frazzoli. Robust Hybrid Control for AutonomousVehicle Motion Planning. Ph.D. Thesis, MIT, 2001.

[5] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," Automatica, vol. 38, no. 1, pp. 3–20, 2002.

[6] Multi-Parametric Toolbox (MPT), for Matlab. http://control.ee.ethz.ch/~mpt/

[7] M. Egerstedt and R. Brockett, "Feedback can reduce the specification complexity of motor programs," *IEEE Trans. Autom. Control*, vol. 48, no. 2, pp. 213–223, Feb. 2003.

[8 G. Fainekos, S. Loizou, and G.J. Pappas, "Translating temporal logic to controller specifications," in *Proc. 45th IEEE Conf. Decision Control*, San Diego, CA, Dec. 2006, pp. 899–904.

[9] R. James Firby. An investigation into reactive planning in complex domains. *Proceedings of the 6th National Conference on AI, Seattle, WA, July 1987,* 1987.

[10] Mitch Ingham, Robert Ragno and Brian C. Williams, "A Reactive Model-based Programming Language for Robotic Space Explorers," *Proceedings of the Sixth International Symposium on Artificial Intelligence, Robotics and Automation in Space: A New Space Odyssey*, Montreal, Canada, June 2001.

[11] Ouimet, M.: The TASM Language Reference Manual, Version 1.1. Available from http://esl.mit.edu/tasm.

[12] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. A Wiley-Interscience, New York, 1994.

[13] Borger, E., Stark, R.: Abstract State Machines. Springer-Verlag, 2003.

[14] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming,* 8(3):231-274, June 1987.