Trajectory Optimization Framework

Johnathan Van Why

September 11, 2013

1 Introduction

This framework is designed for the optimization of trajectories for robotic systems. It can handle underactuated and hybrid systems. In the future, it will have support for multiple "phases", or time periods in which specific dynamics are enforced.

The framework makes use of the MATLAB Symbolic Toolbox to aid in problem formulation and manipulation. The actual optimization, however, is almost always numerical. By keeping the problem in a symbolic form, we allow for the implementation of a mode where the optimization algorithm and problem are formulated together, leading to a final solver optimized for the specific problem's structure.

This document gives a basic description of how to use the framework. It also covers the architectural design of the framework and details on some of the most important algorithms contained within the framework. In addition to this document, code samples are located within the samples/ directory and function as additional documentation for the optimizer.

2 Usage

At the moment, the interface to the optimizer is undergoing a re-design, so it is likely to change in the near future. This section will hopefully be kept up to date with changes.

One thing that currently appears to be invariant is the usage of a central structure, called the "scenario" structure. The scenario structure contains all information necessary to run an optimization – it is incrementally filled out by the optimizer as it completes the various stages of problem setup and optimization. The final scenario structure contains the results of the optimization, and should contain all information that may need to be saved for future use.

The overall workflow for using the optimizer is as follows:

1. Incrementally set up the problem by calling optimizer functions to generate structures.

2. Call traj_optimize() on the scenario structure to do the optimization.

3 Structures

This section contains information on all the values in the structures used to define and optimize the problem. Since the optimizer is built around these structures, correctly setting them up is the majority of the work required to interface with the framework.

Note: This documentation reflects the plans for the next revision of the framework's API.

Each structure contains a struct_type string, which states the "type" of the structure. This allows for error checking and for mixing and matching optimizer functions with structure types.

Additionally, each structure contains a version integer. This version is incremented every time a breaking change is made to one of the structures. All versions are kept in-sync, since the structures contain each other. Each time the version number is incremented, functions for updating all relevant structures should be created for backwards compatibility. The optimizer will automatically update any outdated structures passed in to it. In this way, a scenario structure that is saved at one point in time will remain usable, even if the optimizer is updated.

The ordering of these structures in this section of the documentation reflects the nesting of the structures inside the scenario structure.

3.1 State

This structure represents a dynamic state. It is used in initializing the system dynamics.

Field	Type	Description
names	Cell array of strings	A column array containing the name of each element of the state. Its size matches the size of the overall state.
struct_type	String	The type of the struct – always 'state'
version	Integer	The version number of this structure.

3.2 Input

This structure represents an input to the system. It is used for initializing dynamics and jump maps (when and if jump maps are implemented).

Field	Type	Description
names	Cell array of strings	A column array containing the name of each element of this input. It has the same number of elements an the input it represents.
struct_type	String	This struct's type. Always 'input'
version	Integer	The version number of this structure.

3.3 Scenario

- To be written...