# Preface

This document is designed provide a high level overview and tutorial material for an IOC implementation. The IOC implementation here is the generalized from of Jin et al., a moving window inverse-KKT approach to IOC. The paper can be found here:

@misc{jin2018inverse,

title={Inverse Optimal Control with Incomplete Observations},

author={Wanxin Jin and Dana Kulić and Shaoshuai Mou and Sandra Hirche},

year={2018},

eprint={1803.07696},

archivePrefix={arXiv},

primaryClass={cs.RO},

url={https://arxiv.org/abs/1803.07696}

}

Contents

[Setup the RL model 2](#_Toc37680415)

# Setup the RL model

The backend system responsible for calculating the kinematic and dynamic parameters is an implementation of the Robotics Library, which can be found here:

https://www.roboticslibrary.org/

The Matlab code here already has the RL functions wrapped as Windows MEX DLLs, and can be used as is.

To start, create an XML model outlining how each frame relates to each other. This is most easily done by modifying an existing XML file describing the kinematic chains throughout the model of interest. The XML composes of several components:

* A “frame”, which denotes a coordinate frame. A special type of frame, denoted as a “body”, has non-zero mass, COM, and inertial parameters
* A “transform”, which can be fixed, revolute, or prismatic, associating two frames together by some transform.

Once something is set up, go to ioc/logic and make a copy of ModelRL\_Template.m, which inherits from ModelRL.m. Rename this .m file to more appropriate reflect your dataset, then fill in the appropriate functions outlined in the template.

testModel.m will help you verify that the XML implementation and the ModelRL class is correct.

Once this is completed, update the proper settings in getModel.m

# Setup the IOC model

IOC is ran by loading a json file for its configurations, and performing the IOC. The existing json files can be found in ioc/data\_json. Each dataset will require its own unique json files to run. The json file composes of 3 components:

* “runParamGlobal” which sets the global parameters for all the runs in this file
* Settings set (or not set) by runParamGlobal is then overwritten by “Files”
* Each files can also load a “runTemplate” instead of redefining each set of parameter again.

These settings will instruct the IOC class to load a given type of XML model, the cost functions to use for the IOC, and the trajectory data to load. These parameters are aggregated into the trialInfo (loadTrialInfo.m).

**For this repository, please keep your data and ran results off of github by rerouting all the filepaths to something local. The github repo space is not very big, while this library is capable of creating large amounts of data in a short span of time.**

# View the results

Once the IOC has been ran, go to ioc/Analysis/IOCAnalysis00\_plot.m and update the input/output pathing. The function will then create the individual weight plots, where the weight at time t is calculated from a single window, and the cumulative weight plots, where the weight at time t is the average of every window that overlaps with t.