CSCI 4302/5302: Advanced Robotics
Homework 4 v1: due 28. February, 11:59 p.m. to Canvas.

Learning objectives:

1. Use the `tf` tools in ROS to publish kinematics information and simulate a robotic platform.
2. Test a PID controller on a state stabilization problem.
3. Use a convex optimization and model-predictive control on arbitrary systems, and describe their operation.

Part 1 (2 points).

Complete the following Tutorials:

1. Recording and playing back data
2. Getting started with roswtf
3. Navigating the ROS wiki
4. tf tutorials (C++ or Python): (NOTE: Be sure to do the version of these tutorials for ROS with the CATKIN build system.)
   (a) Introduction to tf2
   (b) Writing a tf2 static broadcaster
   (c) Writing a tf2 broadcaster
   (d) Writing a tf2 listener
   (e) Adding a frame
   (f) Learning about tf2 and time
   (g) Time travel with tf2
5. ROS Parameters

What you turn in to Canvas:

- Upload the following tutorial code and launch files:
  - tf2 broadcaster node
  - tf2 listener node
  - adding a moving tf2 frame
  - adding a tf2 time-travel frame
  - For ROS Parameters: a simple ROS node (in C++ or Python) that reads some parameters on startup and uses them (e.g., printing them out), and
  - a roslaunch file that sets the parameters and runs the C++/Python node.

Part 2 (3 points).

Use `stanley_controller.py` to stabilize a bicycle model of a vehicle to a trajectory. The Stanley controller separates steering and velocity control into two different components, and uses a proportional-integral-derivative (PID) controller for the vehicle velocity. In this Part, you will tune parameters on the PID controller so as to follow the intended path more closely.

What you turn in to Canvas: A page of (4–6) plots of the vehicle's trajectory using various k and k_p parameters. Choose several that you believe demonstrate best the various types of failure modes the vehicle can have if the parameters are chosen poorly, and select the "best" parameters for this problem. You might consider using unconstrained optimization to find the "best." Report your final parameter values.

The current code only includes two control parameters, but **Graduate students: implement an integral and a derivative term into the controller** where it is marked in the Python file and report the same results as above on a second page.

Part 3 (5 points).

Use the Jupyter Notebook `unconstrained_optimiztion.ipynb` to explore concepts of unconstrained optimization as they relate to a model-predictive control problem.

What you turn in to Canvas: A completed Jupyter Notebook.