Introduction to static balancing and whole-body Jacobians

Wyatt Newman
October, 2014

Balancing Atlas involves both dynamic and quasi-static influences. This introduction will focus on quasi-static influences. In brief, if the whole-body center-of-mass, projected onto the ground plane, lies within the convex hull of the foot support(s), then the robot will be statically stable (assuming the joint actuators are capable of freezing the joints at this example pose).

Illustrative code for performing whole-body shifts is contained in the "examples" directory of the class code, in example_whole_body, source files example_main_using_jac_pub.cpp, and example_shift_cg_xdir.cpp.

The source file example_main_using_jac_pub.cpp illustrates a couple of new features. One is using YAML to parse text files to obtain robot data—in this case, joint limits.

The example code also illustrates how to "register" with the low-level joint controller. This involves an initialization transaction, in which the new node informs the LLJC of the potential desire to command joint motions. The LLJC will assign a topic to the new node to be used for this purpose. Subsequently, the new node can "activate" joints, i.e. request "ownership" of these joints for control. The LLJC will accept the request, unless any of the requested joints are already claimed by another node. Correspondingly, it is important that any node issuing joint commands subsequently inform the LLJC when it is yielding control of joints (making these joints available to other controllers). This is the process of "deactivating" joints.

The example code also subscribes to topics published by another node: package "jacobian_publisher", node name "jacobian_publisher." This node uses KDL functions to compute and publish 4 Jacobians: the Jacobians of the left wrist, right wrist, left ankle and right ankle, all with respect to the pelvis frame. For the wrists, the Jacobians are 6x9, since the 3 torso joints also affect the positions and orientations of the wrists. For the ankles, the Jacobians are 6x6.

In the main loop of example_main_using_jac_pub.cpp, the objective is (illustratively) hard-coded to command a pelvis "twist" (a 6x1 vector of three translational velocities and 3 rotational velocities). In this simple case, the twist is set to: [0,-vy,0,0,0,0]. That is, the pelvis is to translate laterally while preserving its x and z displacements and preserving its initial orientation.

The pelvis motion is accomplished by moving the feet with respect to the pelvis. The feet are commanded to move with twist (relative to the pelvis frame) of: [0,-vy,0,0,0,0]. Both feet are commanded to move with the same twist. The specified twist says to keep the feet orientations constant (all rotational velocities = 0, so the feet remain flat on the ground plane). The feet are also commanded to maintain constant z and x displacements, relative to the pelvis, but they are to translate in the y-direction. As a result, there should be no slipping of the feet relative to the floor. The feet will maintain a constant relationship with respect to each other. However, since both feet move along +y relative to the pelvis, and since they do not slip on the floor, the result is that the pelvis will translate along the -y direction.

The foot motions are commanded incrementally using their respective inverse Jacobians.

The illustrative example halts when some joint command is out of range. This may happen, e.g., when

the left leg trys to hyper-extend its knee. Near the straight-leg pose, the leg Jacobian loses rank, and the Jacobian inverse near this situation can result in very large dq commands for a small dy desired. Jacobian singularity (or near-singularity) should be tested before attempting dq motions.

The second example, example_shift_cg_xdir.cpp, shows how to use the leg Jacobians in feedback. This node requires additional support from: package: example_filter, node: foot_wrench_filter. This node will subscribe to atlas/atlas_state, perform low-pass filtering on the foot force/torque sensor values, and republish these on the topics "l_foot_filtered" and "r_foot_filtered". example_shift_cg_xdir subscribes to these topics. It adds the two ankle torques about the y-axis, which corresponds to reaction torques in the sagittal plane (forward/backward leaning). The "twist" command for the pelvis specifies an x-direction velocity intended to null out the ankle torques about y. The result of this is to converge on the robot's center of mass projecting onto the ground plane between the left and right ankles.

This example illustrates how to use sensor data in feedback for whole-body center-of-mass corrections. However, it is not best to place the COM over the ankles. Rather, the COM should project to a point between the centers of the feet (i.e., weight slightly forward of the ankles). Nonetheless, the example illustrates how body motions can respond to force/torque sensors on the feet to make center-of-mass corrections.

The value of the constant "KV" establishes how fast the pelvis should move to null out the reaction torques. This value must be chosen low enough that dynamic influences (accelerations) do not cause the robot to fall. Further, if the ankle torques are too heavily filtered (equivalently, are slow, or phase lagged, or time delayed), the reaction in terms of pelvis corrections must be slow enough that the time-delayed sensor signals do not cause instability. (This may motivate re-turning the filter constants).

"Quasi-static" motions are motions for which stability is dominated by statics, i.e. inertial effects are negligible. If the pelvis corrections are too fast, the quasi-static assumption will be violated, and the robot may fall. On the other hand, if CoM corrections are made too slowly, the robot may fall down before balance is established. Tuning the pelvis response gains can be challenging. Faster responses can be achieved with additional consideration of inertial effects, although the equations become more complex.

To run the example code (see the README file):

- roslaunch hku_worlds beer_table.launch (or some alternative world)
  (in drcsim v4: roslaunch drcsim_gazebo drc_practice_task_6.launch)
- rosrun jacobian_publisher jacobian_publisher
- roslaunch fc_bringup sim_task_4.launch
- rosrun sim_controller USER (this will run for 3 seconds)

and finally, the example whole-body node:

- rosrun example_whole_body example_main_using_jac_pub

For the 2nd example, also run:

- rosrun example_filter foot_wrench_filter

and (instead of example_main_using_jac_pub):

- rosrun example_whole_body example_shift_cg_xdir