Computation of Gravity Loading

Wyatt Newman

Oct 22, 2013

Computation of gravity loading is important both for accurate position control of hands as well as for controlling balance. For Atlas in particular, the arms are quite heavy. With an arm outstretched, the downward tug of gravity on the hand and relatively heavy wrist, forearm and elbow joints presents a substantial load on the shoulder joints. If the joints are controlled by proportional plus derivative controls, then there will be an angular position error such that in equilibrium:

Kp*(q_desired – q_actual) = trq_gravity

where Kp is the proportional gain on a joint, q_desired is the desired angle of this joint, q_actual is the actual angle of this joint in static equilibrium, and trq_gravity is the torque of the joint required to oppose gravity loading. For a shoulder joint in particular, the gravity loading, trq_gravity, can be substantial, resulting in significant angular error, q_desired – q_actual. At the same time, the resulting hand error will be proportional to the angular error and the distance from the shoulder joint to the hand. With the arm outstretched, the gravity loading is maximized, as well as the distance from shoulder to hand, resulting in what can be substantial position error, often referred to as "gravity droop." The gravity droop will be reduced with larger proportional gain, Kp, but there are limits to how large this can be before the joint control becomes unstable.

Gravity-droop errors can be reduced by compensating for gravity torques. If the loading due to gravity can be computed, then this torque can be added as a command to the respective joint. The modified control law then becomes:

trq_ctl = Kp*(q_desired-q_actual) + Kv*(qdot_desired – qdot_actual) + trq_gravity(q_vec)

In the above, trq_ctl is the control torque to be commanded to a given joint. q_vec is a vector of all of the joint angles. qdot_desired is the desired joint velocity, and qdot_actual is the actual (sensed) joint velocity. trq_gravity(q_vec) is the computed joint torque required to maintain the system in static equilibrium at the pose q_vec considering gravity loading.

A convenient expression for computing gravity loading is the center of mass of the system. Each link of our robot has its own mass, m_i, and center of mass. Using forward kinematics and a mass-property model, we can compute the location of center of mass "i" as **r**_i. Conveniently, we can compute each one of these vectors with respect to the pelvis frame: **r**_i/pelvis. The total mass of the system, M, is:

eqn 1: M = summation_i { m_i }

The center of mass of the combined system, **c**, can be expressed as:

eqn 2: **c** = [summation_i **r**_i*m_i]/M

The point in space "**c**", as well as each link's center-of-mass location **r**_i, has meaning independent of any frame of reference. However, for computing numerical results, we will need to specify a frame of reference. For many computations, the pelvis frame is convenient to use. The center of mass with respect to the pelvis frame can be denoted as: **c**/pelvis, for which consistent computations would use individual link center-of-mass locations expressed in the pelvis frame, **r**_i/pelvis.

Arm gravity torque compensation: If we consider the torso to be held fixed, then the position of

the right hand will be a function of the right-arm joints, which, for Atlas, consists of 6 joints. We can consider a center of mass of just the right arm, **c**_ra, which is computed identically to Eqn 2, except that the index of summation runs through only the joint numbers associated with right-arm joints. The sum of right-arm masses is M_ra, computed identically to Eqn 1, except indexing only through the right-arm joints.

We refer to “gravity torques” as the joint torques required to achieve static equilibrium with respect to gravity loading. We can compute these torques using a Jacobian approach and an energy argument. Consider moving a single joint, q_i, by some perturbation, dq_i. In this thought experiment, the joint is to be moved slowly, such that inertial and damping effects are negligible, and the resulting joint torque, Q_i, is thus in quasi-static equilibrium with the gravity torques. Performing the test move requires that the i'th motor perform some work: Q_i*dq_i. This incremental work injection may be positive or negative. Since the kinetic energy remains nearly zero, and since we have not considered frictional losses, the work input must balance an increase (or decrease, if work is negative) in potential energy. The potential-energy increase would be realized by some net increase in elevation of the various link masses. Since we care about only changes in the elevation (motion relative to the direction of gravity), we need to express the center of mass with respect to some frame aligned with gravity. We will refer to a “world” frame for which the z-axis is parallel to the direction of gravity. For this frame, the location of the origin is irrelevant, provided this origin is stationary with respect to the world. In the present case, if we freeze the torso and consider right-arm movements, a convenient choice of world-frame origin is the origin of the pelvis. We can transform coordinates with respect to the pelvis into coordinates with respect to our defined “world” frame by the transform: **r**_i/w = **R**_p/w*__r__i/pelvis. That is, if we know the orientation of the pelvis frame in the world, expressed as the rotation matrix **R**_p/w, then we can transform a vector expressed in the pelvis frame, e.g. **r**_i/pelvis, into a vector expressed in our world frame, **r**_i/w , by pre-multiplying by the orientation of the pelvis frame. In establishing **R**_p/w, since we will only care about changes in elevation for the present computation, we only really require that the world frame have its z axis parallel to gravity, whereas the x and y axes may be at any convenient yaw angle. Such a transformation is available on Atlas (and in drcsim) via the IMU (Inertial Measurement Unit) sensor values. An “orientation” is published on the topic /atlas/atlas_state as a quaternion. This quaternion may be converted to a 3x3 rotation matrix to be used in the above transformation.

Using forward kinematics, we can compute the location of each link's center of mass with respect to the pelvis, **r**_i/pelvis. This can be done conveniently using the ROS package “KDL” (Kinematics and Dynamics Library). Each of these vectors can be transformed to our gravity-aligned world frame using **R**_p/w, which is available via IMU measurements. As a result, we can compute the center of mass of the right arm in our defined world frame, **c**_ra/w, as:

Eqn 3: **c**_ra/w = [summation_i_ra **r**_i/w m_i] /M_ra

In the above, the summation of “i_ra” is over all joints of the right arm only. M_ra is the summation of all right-arm masses.

The potential energy of the masses of the right arm, U_ra, is equal to:

Eqn 4: U_ra = sum_i_ra m_i*g*z_i
= sum_i_ra m_i*g*__r__i/w*[0;0;1]
= M_ra*g*__c__ra*[0;0;1]

For incremental input of work, the resulting incremental change in potential energy due to a perturbation of the j'th joint angle, dq_j, will be:

Eqn 5: d U_ra/dq_j = [0 0 1] * M_ra*g*(d **c**_ra/dq_j)
The change in potential energy depends on the derivative d **c**_ra/dq_j, which is a column vector, 3x1. If we compute the corresponding column vector for each joint of interest (e.g., all right-arm joints), we can assemble these column vectors into a Jacobian matrix, **J**_cra = d **c**_ra/d**q**. This computation requires knowledge of a collection of link center-of-mass Jacobians:

Eqn 6: **J**_cra = sum_i_ra d **r**_i/d**q** = sum_i_ra **J**_cmi

Each of the component Jacobians can be computed conveniently using the KDL package. These individual Jacobians can be added, term by term, to produce the net center-of-mass Jacobian, in this case for the right arm, **J**_cra/w, where "/w" means that the output coordinates are expressed with respect to our defined world frame.

We care in particular about changes in the elevation. We can strip off the z-component by the dot product with a unit vector in the z direction, [0,0,1].

Further, instead of considering each joint individually, we can consider all right-arm joints collectively in the vector **q** and we can compute the net change in elevation for a perturbation of d**q** as:

Eqn 7: dU_ra = [0, 0, 1]*M_ra*g* **J**_cra/w* d**q**

At the same time, the work input from the right-arm motors for this perturbation is:

Eqn 8: dW = **Q**'* d**q**

where **Q** is a column vector (and **Q**' is its transpose, a row vector) of the right-arm joint torques.

Equation work input with change in potential energy, we get:

Eqn 9: **Q**'* d**q** = [0, 0, 1]*M_ra*g* **J**_cra/w* d**q**

and from this we can infer that the gravity torques on the right arm must equate to:

Eqn 10: **Q**'= (M_ra*g)* [0, 0, 1]***J**_cra/w

which is the weight of the right arm (M_ra*g) times the third row of the right-arm center-of-mass Jacobian.

To compensate for gravity droop of the right arm, the controller should add this set of gravity torques to the the existing controller (e.g., proportional-plus-derivative controller).

Whole-body center of mass computations: For performing balancing, we will need to extend the above arguments to whole-body center-of-mass computations, as per Eqn 2. Consider trying to balance on one foot (say the right foot). For our potential-energy computations, we can no longer assume that the pelvis origin will be fixed with respect to the world. (Perturbations of right-leg joints will change the pelvis origin with respect to the world). Instead, we can consider computing the center of mass with respect to a new frame with origin at the right ankle. The orientation of this world frame will still have its z-axis aligned parallel to gravity. It may be convenient to align the world-frame x and y axes along the right-foot major and minor axes.

KDL and Atla's IMU orientation sensing can again be used to compute all terms **r**_i/w, but in this case we will define the world frame to have origin at the right ankle, and the index "i" will run over all joints of the robot.

We can compute a corresponding whole-body center of mass, **c**_robot/w. We can also compute a center-of-mass Jacobian as **J**_cRobot/w = d**c**_robot/w / d**q**. The total mass of the robot is M_robot. The change in potential energy of the robot for a perturbation d**q** is:

Eqn 11: dU_robot = (M_robot*g)* [0, 0, 1]***J**_cRobot/w* d**q**

The incremental work performed for this perturbation is **Q**'* d**q**. From conservation of energy, we can deduce:

Eqn 12: **Q**'= (M_robot*g)* [0, 0, 1]***J**_cRobot/w

The above assumes that the only body part in contact with the world is the right foot. Under the assumption that the right foot does not slip, there is no work performed due to torques or forces on the right foot. If other body parts are in contact with the world (e.g., left foot touches, or left or right arms touch a railing or a wall), then these additional contact forces/torques must be considered to compute balancing.

Note that Eqn 12 considers all joints of the robot, including the subset of right-arm joints considered separately earlier. If the right foot were glued to the floor, then the joint torques of Eqn 12 would result in static equilibrium with gravity loading. However, if we desire that the robot balance on the right foot, another consideration is required. Specifically, for the world frame defined with respect to the right ankle, the robot's center of mass, **c**_robot/w, must have x and y components that lie in the interior of the footprint of the right foot. If the center of mass projection onto the horizontal plane is a point that lies outside the boundaries of the foot, then the foot will be incapable of producing the moments necessary to oppose gravity. Although the ankle may be strong enough, the foot would simply tilt about its edge and the robot would fall down. It is therefore a consideration of balancing that the COM of the robot be moved to a point within the boundary of the stance foot.

To achieve balancing on one foot, one may start with balance on two feet (which is easier). The COM of the robot may be shifted to lie above the chosen (e.g. right) ankle using Jacobians. From the perspective of the pelvis, the robot can be commanded to move its foot with specified dx, dy, dz while preserving its orientation with respect to the pelvis. Each leg is controlled by 6 joints, with respect to the pelvis. Defining desired foot motion of fixed orientation (dTheta_x, dTheta_y, dTheta_z) = (0,0,0) and specified displacement (dx, dy, dz), a 6x1 Cartesian motion is defined. Each leg has 6 joints from the pelvis down to the foot. Thus, there is a 6x6 Jacobian that relates incremental motions of the foot to incremental perturbations of a leg's joints.

As an illustration, if the simulated robot is "pinned" at its pelvis, and the knees are slightly bent (thus elevating the feet off the ground), one can command a coordinated motion of both feet with respect to the pelvis such that their orientations remain fixed and their translations preserve their relative positions. For example, if the soles of the feet are coplanar to each other and coplanar to the ground, and if the feet are parallel to each other at a given separation distance, a coordinated Jacobian move specifying the same dx, dy, dz and zero orientation change would result in the feet moving such that: foot soles remain coplanar, foot soles remain parallel to the ground, feet remain parallel to each other, and feet retain a constant separation from each other. Equivalently, if the pelvis were not pinned, but the feet were on the ground, the pelvis would move by -dx, -dy, -dz and the feet would not slip relative to the ground. Thus, leg Jacobians can be used to compute incremental leg-joint motions to move the robot's center of mass with respect to the feet. This can

be used to center the robot's weight over one ankle prior to attempting to stand on one leg.

Extending this logic to balancing on two feet requires some additional derivations. Part of the complication is that two feet in contact with the ground results in a closed-chain structure, for which there is ambiguity with respect to internal forces. For example, the two feet could have a force tending to separate them (to do the “splits”), but if the Coulomb friction is high enough, there would be no motion. Similarly, additional ankle torques and could be exerted that cancel each other. There is thus no unique solution to the closed-chain case. However, simplifying assumptions can be introduced to solve for a viable set of forces/torques, but this analysis is deferred for the present.