Modeling the Atlas Robot:

Wyatt Newman

September, 2014

These notes provide a quick introduction to the "Universal Robot Definition File" (URDF), specifically referencing the Atlas robot. References here are with respect to the Atlas model of DRCsim version 2.7. Updates are expected with significant changes.

Robot URDF Model File:

A model of the Atlas robot, including inertial properties, visual properties, collision properties, and kinematic properties, is needed for Gazebo simulations. This model should be as accurate as possible in matching the actual Atlas robot. At the time of this writing, there are still significant differences in kinematics and mass properties, but future releases will improve the modeling fidelity. More radical changes to the model are expected with anticipated new arms and additional battery pack, presumably Fall 2014.

When launching a Gazebo simulation, the launch file specifies where to find the robot URDF file. For the hku launch files, we direct gazebo to use a robot description in the directory:
`catkin/src/hku_common/hku_robot_description/urdf`
in which the urdf file is: `/atlas_v3_calibrated.urdf`

The URDF describes 28 movable joints of the robot. (A separate URDF is loaded to describe the 12 joints each in the right and left hands, and another file for the sensor head). Sensor frames are also defined for the situational cameras. A description of the URDF format can be found at: http://www.ros.org/wiki/urdf/Tutorials.

An abbreviated example of link and joint specifications, extracted from the `atlas_v3_calibrated.urdf` file, appears below:

```
 <link name="utorso">
    <inertial>
      <mass value="52.007" />
      <origin xyz="-0.0923 0 0.3" rpy="0 -0 0" />
      <inertia ixx="1.466" ixy="0.00362" ixz="0.336" iyy="1.51" iyz="0.001" izz="1.3" />
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 -0 0" />
      <geometry>
        <mesh filename="package://atlas_description/meshes_v3/utorso.DAE" scale="1 1 1" />
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 -0 0" />
      <geometry>
        <mesh filename="package://atlas_description/meshes_v3/utorso.DAE" scale="1 1 1" />
      </geometry>
    </collision>
    <visual>
      <origin xyz="0 0 0" rpy="0 -0 0" />
      <geometry>
        <mesh filename="package://atlas_description/meshes_v3/utorso_pack.DAE" scale="1 1 1" />
```

```
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 -0 0" />
      <geometry>
        <mesh filename="package://atlas_description/meshes_v3/utorso_pack.DAE" scale="1 1 1" />
      </geometry>
    </collision>
  </link>
```

These extracted lines define the link “utorso”. Each link has an associated coordinate frame. The link’s mass and its center of mass location is specified relative to the link’s coordinate frame. E.g., for utorso, the mass is 52kg, and the center of mass is located at (x,y,z) = (-0.09, 0, 0.3), as measured in the link’s reference frame. The rotational inertia for this link (with units kg-m^2, rounded to 2 digits) is :

$$H = \begin{bmatrix} 1.5 & 0.003 & 0.34 \\ 0.003 & 1.5 & 0.001 \\ 0.34 & 0.001 & 1.3 \end{bmatrix}$$

The off-diagonal terms indicate that the inertia is not specified with respect to principal axes. Rather, the moments of inertia are expressed with respect to a frame with origin at the center of mass and axes oriented by a roll-pitch-yaw description relative to the link’s reference frame. For the utorso example, the inertial-frame orientation is rpy="0 0 0", i.e. oriented coincident with the link’s reference frame. (If the rotational inertia matrix had been diagonal, this would have indicated that the principal axes are coincidentally aligned with the link-frame axes).

Link specifications have additional properties, including visualization properties (the display appearance) and collision properties (typically a solid model, such as one or more cylinders, that is simpler to check for collisions than the more detailed visual model). The utorso example specifies a file that contains a mesh model for graphical display of the utorso link. For this link, the collision model is identical to the visualization model.

A kinematic chain of links is defined by specifying joints that connect links. The utorso link is a “child” of link mtorso and is the parent of link l_clav. The snippet below from the URDF specification declares that utorso is a child link of mtorso, and that these two are related through a revolute (hinge) joint. The name of the joint that constrains these two links is “back_ubx”. The angle of this joint, plus the frame information below, is sufficient to define how the utorso coordinate frame is related to the parent’s (mtorso) coordinate frame.

```
  <joint name="back_bkx" type="revolute">
    <origin xyz="0 0 0.05" rpy="0 -0 0" />
    <axis xyz="1 0 0" />
    <parent link="mtorso" />
    <child link="utorso" />
    <dynamics damping="0.1" friction="0" />
    <limit effort="200" velocity="12" lower="-0.697" upper="0.572" />
    <safety_controller k_position="100" k_velocity="100" soft_lower_limit="-10.6981" 
soft_upper_limit="10.6981" />
```

The child link’s frame is defined such that its origin lies on the parent/child joint axis. E.g., the origin of the utorso link lies on the rotational axis of the back_bkx joint (previously called the back_ubx joint). It is also necessary to define the orientation of the child frame relative to the parent’s frame. This specification seems ambiguous in the URDF description, but the apparent intent is that this orientation (given by: rpy="0 -0 0" for the current example) is the orientation of the child frame relative to the parent frame when the value of joint-angle rotation of back_bkx is zero. (Clearly, the child linkframe changes relative to the parent frame as a function of joint angle for the connecting joint, so there are no static values of RPY that specify this relationship in general).

Although the child-link’s frame has an origin that lies on the parent/child connecting joint axis, the parent/child joint axis is not necessarily conveniently aligned with any of the frame axes. For the case of the back_bkx joint, the joint orientation is given by: <axis xyz="1 0 0"/>, which is conveniently aligned with the x-axis of the child (utorso) frame.

The URDF snippet below describes the joint connecting link utorso to link r_clav via joint “r_arm_shy.”

```
  </joint>
  <joint name="l_arm_shy" type="revolute">
    <origin xyz="0.06441 0.13866 0.10718" rpy="0 -0 0" />
    <axis xyz="0 0.5 0.866025" />
    <parent link="utorso" />
    <child link="l_clav" />
    <dynamics damping="0.1" friction="0" />
    <limit effort="212" velocity="12" lower="-1.474" upper="0.769" />
    <safety_controller k_position="100" k_velocity="100" soft_lower_limit="-11.5708"
soft_upper_limit="10.7854" />
  </joint>
```

For the case of the r_arm_shy joint, the joint is oriented as: <axis xyz="0 0.5 -0.866025"/> , which defines a unit direction vector, specified in the child-link’s frame. Note that although the child link’s frame rotates relative to the parent link, the joint-axis expressed in the child-link frame remains valid. (The direction of the parent/child joint axis could also have been expressed in the parent frame, and this specification would also be constant as the robot moves. The choice to express it in the child frame appears arbitrary, and unfortunately inconsistent with specification of the “origin” values, which are expressed in the parent frame).

Most of the joints in the Atlas model are aligned with one of the child-link frame axes. The clavicle is a notable exception, as this joint is tilted 30-degrees relative to the “spine” (z-axis of the utorso frame).

In addition to specifying kinematic constraint properties between two joints with a connecting link, the joint specification also includes actuator information for the joint. Range-of-motion limits (lower="-1.474" upper="0.769" for r_arm_shy) specifies a minimum and maximum angle of the range of motion (in radians) . The corresponding actuator has a maximum torque of effort="212" N-m. The joint also has viscous friction, defined as damping="0.1" N-m, but Coulomb friction has been set to zero. The actuator also has a velocity limit of velocity="12" rad/sec. A safety controller is defined to help prevent hitting angle hard stops at high speed (though this is not implemented on physical Atlas). Note that the true values of motion limits, actuator saturation, mass, inertia, etc, need to be better tuned to the actual

Atlas robot. Inertial properties are expected to change significantly with the addition of batteries and the retrofit of new arms.

Joint Actuator Effort Specifications: At the URDF level of robot modeling, actuator inputs are assumed to be joint torques (for revolute joints). However, the atlas_command message provides the user with joint-angle command capability. The missing layer between these two levels of modeling is a set of joint feedback controllers. The Atlas robot (and the corresponding drcsim simulator) performs joint-by-joint actuator control with linear feedback. For the physical robot, this is performed within a computer in Atlas' chest at 1kHz.

Although the user cannot change the actuator feedback algorithms within Atlas (or within drcsim), the parameters of these algorithms are user specified. All control gains are set to values specified in atlas_command message each time a new command is sent. The command message is flexible enough that the user can command torques directly from a ROS node and can set all feedback gains to zero, if desired.

On the physical Atlas robots, the actuators have dynamics very different from the URDF specification. Physical Atlas has electro-hydraulic actuators, and the joints have significant Coulomb friction. The HKU code has tuned values for Atlas' feedback algorithm that can approximately accommodate torque commands (which is useful, e.g., for gravity-load compensation). Further, the HKU code includes an option for compliant-motion control. Although the feedback implementation of compliant-motion control on Atlas is complex, the result is that Atlas can emulate the behavior of relatively soft torsional springs at his joints with relatively low apparent Coulomb friction. In this control mode, physical Atlas behaves similar to drcsim Atlas. However, the interface to this mode still consists of defining "attractor" angles (equivalent to joint position commands) and specifying (typically soft) joint stiffnesses (equivalent to K_position gains). This control mode is useful for compliant-motion control, e.g. for interacting with doorknobs, doors, valves, tools, etc. It is up to the programmer to design how to take advantage of such compliance.

Conclusion: The URDF model for Atlas in DRCsim provides the necessary information to simulate kinematics and dynamics of Atlas, including gravity effects, inertial effects, influence of actuator torques, and influence of contact forces. To make the simulator a realistic model of Atlas, further refinement of dimensions, mass properties, joint limits and actuator properties will be required. The current model is nonetheless suitable for much code development, and improvements to the model will make it increasingly valuable.