Final Project Report
ME/CS/EE 133a
Fall 2021

By:
Gilbert Bahati, Sergio Esteban

Instructor: Günter Niemeyer Ph.D.

California Institute of Technology

Atlas The Drummer

Gilbert Bahati, Sergio Esteban

December 10, 2021

Contents

1 Introduction

The idea of this project is to have the humanoid Atlas robot play on a drum set in Rviz. The focus of this project was rooted in trajectory generation for manipulators. In this case, the manipulators are Atlas' arms. An algorithm was developed that takes a desired sequence of drum "hits" and generates trajectories such that Atlas hits each drum member at the correct time. Additionally, a secondary task was chosen to make Atlas dance. This project utilized several topics from ME 133a.

2 Robot and System Description

2.1 ATLAS Robot

Atlas is a humanoid robot made by Boston Dynamics. In this project, an early version of the robot was used. This robot has 28 degrees of freedom (DOFs), however, only 17 DOFs were used since the DOFs in the legs were in a fixed position. Degrees of freedom include 3 DOFs from the torso and 7 DOFs from each arm. In total, 17 DOFs are used. Given Atlas' human-like attributes, the work space in this project is akin to that of a human's arm work space. The picture below summarizes the joints of interest:

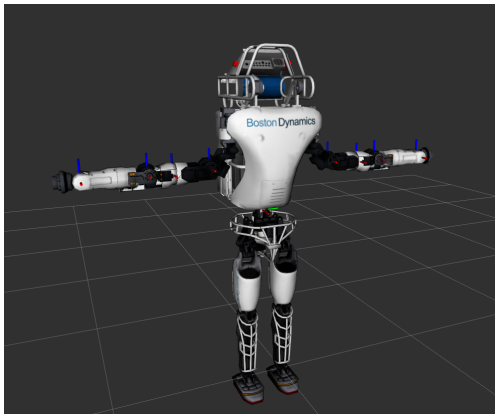


Figure 1: Atlas Standing.

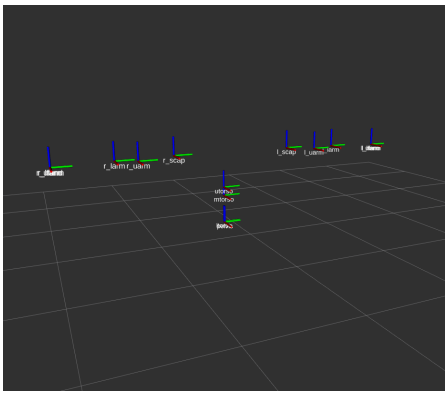


Figure 2: Torso, left and right arm DOFs of interest.

2.2 Rviz

Atlas was setup in Rviz along with drum sticks, a drum set, and a chair (Figure 4). The drum sticks and chair were created in Solidworks, converted to STL meshes, and imported into Atlas' Unified Robotic Description Format (URDF) file as links with fixed frames (Figure 12).

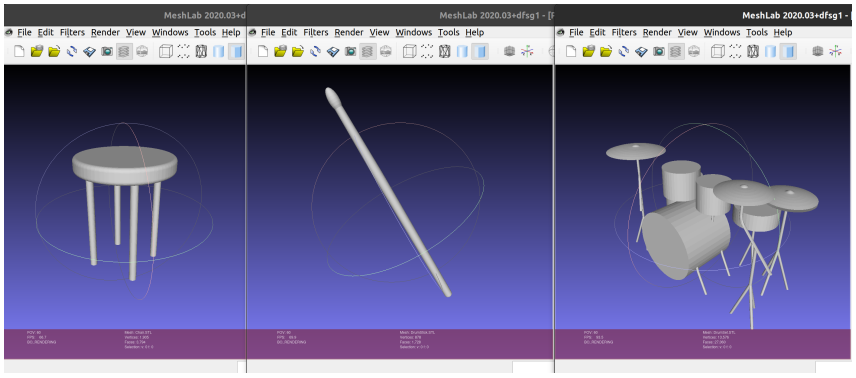


Figure 3: The stick and chair were modeled in Solidworks while the drum set is from an open source site: https://grabcad.com/library/drum-kit-2.

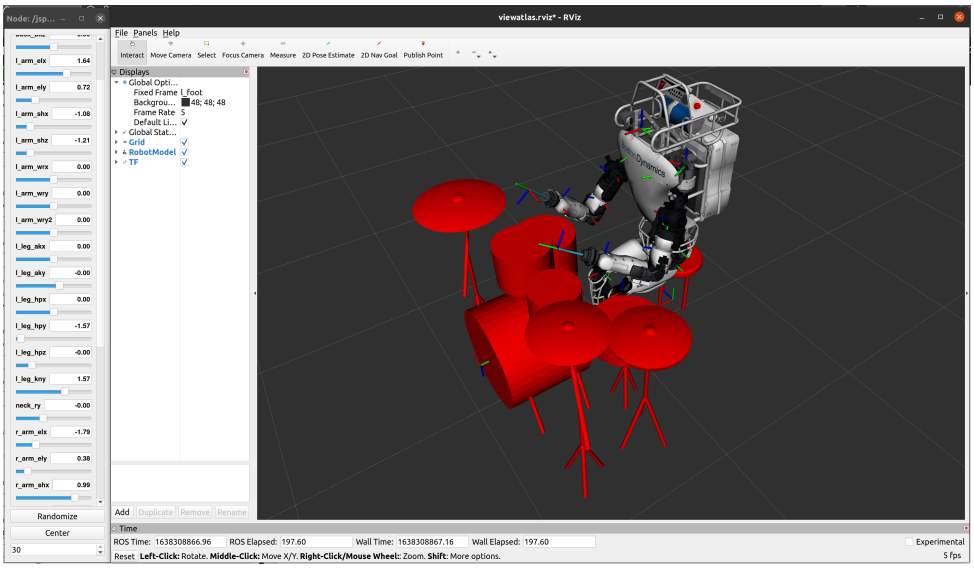


Figure 4: Rviz setup with Atlas sitting and ready to play the drums.

3 Tasks

For trajectories, the task space was used to generate desired trajectories for each arm. That is, Atlas has to strike a drum component with the drum stick at a specified tempo/beat, location, and orientation.

3.1 Drum Set Layout

The drum layout (Figure 5) shows the layout of the drum kit. Member 1 is the high hat, members 2,7 are the cymbals, members 4, 5, 6 are the toms, and lastly, member 3 is the snare. Ultimately, the objective is to have Atlas hit a combination of these members to make a drum beat.

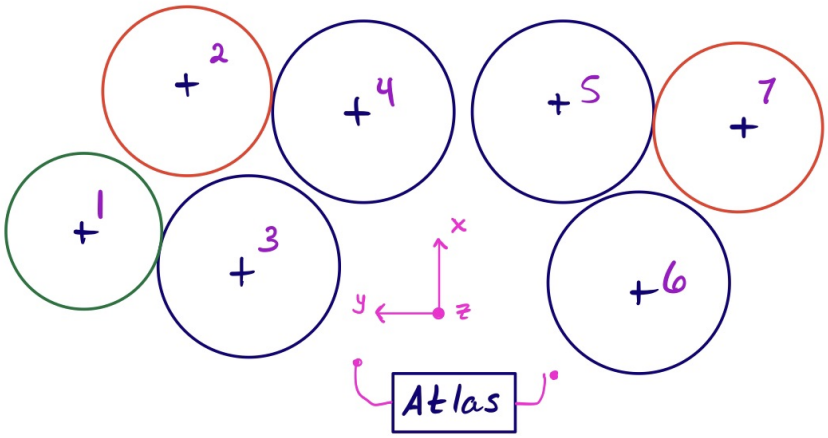


Figure 5: Drum kit layout.

3.2 Rhythm

Most songs have a 4/4 time signature. This means that each measure of a song consists of 4 beats (4 quarter notes). In this project, the duration of a beat is equivalent to the time it takes the drum stick to travel from one drum member to another. In music, the speed (pace) of a song is know as its

tempo, usually in units of beats per minute (BPM). To construct a rhythm with Atlas, we used the BMP parameter to determine beat lengths, b in seconds:

$$b = \frac{60}{BPM}$$

In the demonstration, Atlas jams along with *The Gap Band*'s "Outstanding". The song's 98 BPM tempo was used to find the beat length.

4 Algorithm and Implementation

4.1 Jacobian Matrix

For this system, the legs are kept at fixed angles. Therefore, the `pelvis` is set as the origin and our kinematic chains are constructed from there. However, the torso "branches off" into two arms (Figure 12). This certainly posed problems, which were solved by constructing a large Jacobian matrix that handled the two different arm "branches". This resulted in 3 DOFs (pelvis, lower torso, upper torso) attached to two arm branches that each have 7 DOFs. This gave rise to the following large Jacobian matrix (i.e., 17 DOF system):

$$\dot{x} = \begin{bmatrix} \dot{x}_{left} \\ \dot{x}_{right} \end{bmatrix} = \begin{bmatrix} J_{lback} & J_{larm} & 0 \\ J_{rback} & 0 & J_{rarm} \end{bmatrix} \begin{bmatrix} \dot{q}_{back} \\ \dot{q}_{larm} \\ \dot{q}_{rarm} \end{bmatrix} \tag{1}$$

4.2 Secondary Tasks for Singularities

In order to ensure Atlas avoided singularities, we fixed some desired orientations as secondary tasks to ensure smoother (human-like) trajectories. For example, we encountered singularities when stretching the arms too far wide (Figure 6).

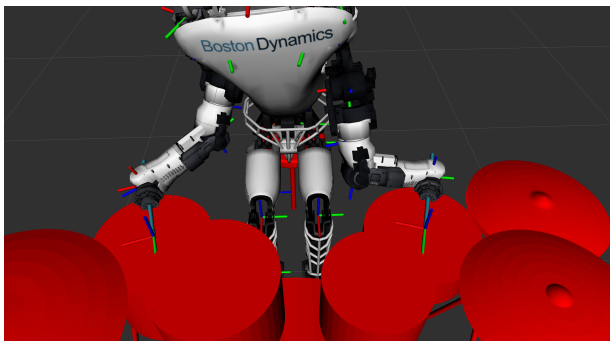


Figure 6: Atlas reaching singularities.

In order to avoid issues like these, we:

- Ensured the initial orientations were conducive for Atlas not to hit singularities (Figure 6).
- Centered the elbow joint angles around desired values in order to avoid far stretches.
- We fixed the back/torso orientation to have Atlas sit upright while playing the drums.
- In conjuction with the above two points, we leveraged the weighted pseudo-inverse (2) to compute $\dot{q}$ and adjusted γ to the ensure conservative behavior around singularities.

$$J_w^+ = (J^T J + \gamma^2 I_N)^{-1} J^T \tag{2}$$

$$\dot{q} = J_w^+ v_p + (1 - J_w^+ J)\dot{q}_{secondary} \tag{3}$$

Note: A coherent summary for the above process is included in the pseudo code (Section 4.5)

4.3 Secondary Tasks for Dancing

We furthermore implemented an extra secondary task to ensure Atlas dances along with the music. This is shown in the accompanying video. This secondary task is active when Atlas' torso periodically swings from side to side during the video - this happens in the beginning of the video and about 3 times during the video. More specifically, Atlas has its arms resting on a drum head and dances with its torso.

We achieved this by encoding "Hold" segments (when a pause is needed) and using a path variable (as described in Section 5.1: Particular Features), to encode a rotation about the z axis for the torso to swing from side to side).

Note: As in the previous section, a coherent summary for the above process is included in the pseudo code (Section 4.5)

4.4 3D Splines and Rotations

We characterized our position splines in the *task space* and tip rotations via a *path variable*. We describe how we achieved this in detail below.

4.4.1 Position Splines

The stick model was added to the `atlas_v5.urdf` file and the tip of the stick was given a fixed frame (see Figure 12). When sticks are in the hands of a human drummer, they tend to have parabolic trajectories when traveling from one drum member to another. Given this observation, a parabolic trajectory was desired for the trajectory of the sticks in the Atlas drummer model. This was achieved via two linear equations and one parabolic equation. Note: These segments were created in the *task space*.

It is important to know the world frame in Rviz. The x-y plane is the "floor" in Rviz and the z axis is pointing upward and normal to the x-y plane. To get from some arbitrary initial point p_o to some arbitrary final point p_f, three separate spline equations were derived. The splines are functions of time, t, and drive the stick from p_o to p_f in some arbitrary amount of time, T (in this case, b which is extracted from a song's tempo in BPM). This whole process is one *segment* of the total chain of drum strikes.

$$p(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \quad , \quad 0 \leq t \leq T \tag{4}$$

Note only the x and y trajectories need to be linear. Therefore, x and y are simply linear equations (see Figure 8b).

$$x(t) = \left(\frac{x_f - x_o}{T} \right) t + x_o \tag{5}$$

$$y(t) = \left(\frac{y_f - y_o}{T} \right) t + y_o \tag{6}$$

Now, while the x and y trajectories are linear, the z trajectory must be parabolic due to the aforementioned details (see Figure 8a). Therefore, we start with the skeleton of a standard quadratic equation and constrain its three degrees of freedom (a,b,c):

$$z(t) = at^2 + bt + c \tag{7}$$

with these constraints:

$$\text{at } t = 0: \; z(0) = z_o$$
$$\text{at } t = \frac{T}{2}: \; z\left(\frac{T}{2}\right) = z_h + h$$
$$\text{at } t = T: \; z(T) = z_f$$

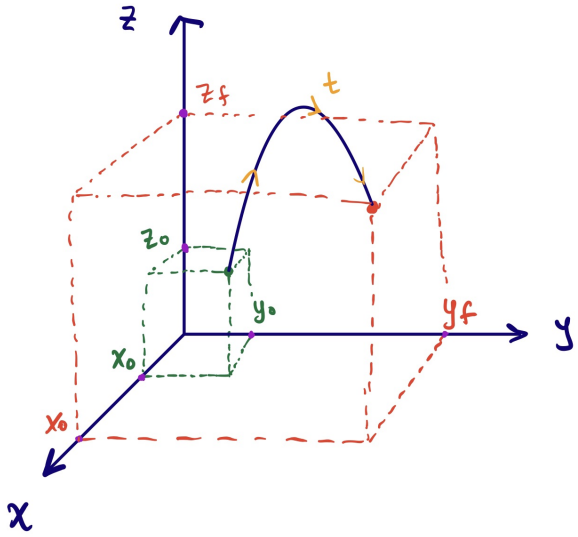
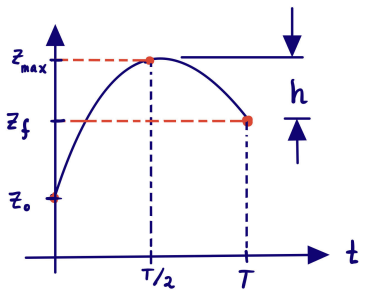


Figure 7: 3D spline illustration.

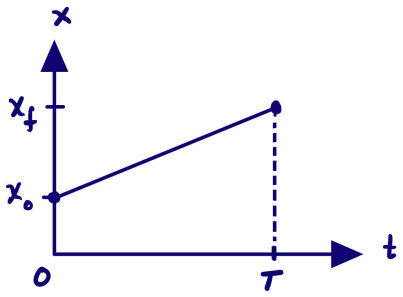
Here, z_h is the greater of the two z values, $z_h = \max\{z_o, z_f\}$ and h is a variable named *height factor* that controls the difference between the apex of a parabola and the highest z-coordinate in between drum hits. After applying constraints, we get the coefficients for Equation 7:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} T^2 & T \\ \frac{1}{4}T^2 & \frac{1}{2}T \end{bmatrix}^{-1} \begin{bmatrix} z_f - z_o \\ z_{max} - z_o \end{bmatrix} \quad , \quad c = z_o \tag{8}$$

It is required to guarantee that the 4×4 matrix in Equation 8 is invertible. Note that this 4×4 matrix is invertible $\forall\, T > 0$.



(a) Quadratic spline for $z(t)$.



(b) Linear spline example for $x(t)$, $y(t)$, and $\theta_z(t)$.

4.4.2 Tip Rotations

In addition to characterizing the drum stick location, the tip orientation was characterized. Rotation was applied to the drum stick about the z-axis (pitch) and the positive x-axis (yaw). This rotation matrix is a function of a *path variable:* s that ranges from 0 to 1 (scaled using the T from before: $\frac{0}{T} \rightarrow \frac{T}{T}$) for every segment of the total chain of drum strikes:

$$R(\theta_z, \theta_x) = R_z(\theta_z(s))R_x(\theta_x(s)) \tag{9}$$

For every segment, the the tip rotates about the z-axis linearly and about the x-axis with a sinusoidal motion.

$$\theta_z(s) = \left(\frac{\theta_{zf} - \theta_{zo}}{T}\right) s + \theta_{zo} \tag{10}$$

$$\theta_x(s) = \theta_A \sin(\pi s) \tag{11}$$

Here, $\theta_z(s)$ takes some initial angle θ_{zo} and terminates at some final angle θ_{zf}. The equation for $\theta_z(s)$ is similar to Equation 5. Notice that θ_x starts and ends at 0 rad, but finds it's peak at $s = 0.5$ at some amplitude θ_A.

The results to the above are shown in detail in section 5.1.

4.5 Algorithm Implementation

All algorithms were realized via Python code. Algorithm 1 shows a top level implementation.

Algorithm 1 Atlas Trajectory Generator (Playing Drums)

```
Initialize ROS and all dependencies
Create kinematic chains                          ▷ Torso to left arm, Torso to right arm
Create task space segments                       ▷ Spline Trajectories
Create path variable s segments
Initialize joint angle in conducive orientation
procedure UPDATE(simulation timestep)
    Evaluate task space segment
    Evaluate path variable segment
    while ROS is running do
        if Current task space segment type is "Hold" then
            Encode Atlas torso dance moves as a secondary task using path variable
        else if Current task space segment type is "Goto" then
            Fix Atlas torso/back to upright orientation as a secondary task
            Compute desired tip orientations as a secondary task using the path variable
        Fix/center elbow orientations            ▷ must always be in elbow down orientation
        Compute Jacobians
        Compute orientation errors
        Compute q̇ = J_w^+ v_p + (1 − J_w^+ J) q̇_secondary
        Update joint angles
        Publish angles to rostopic
```

5 Particular Features

5.1 Realistic Drum Strikes

One of the features that we are somewhat proud of is realistic drum striking (see traced trail below). We were able to do this by implementing what was described in Section 4.4.2 ie. adjusting our tip rotations with respect to the path variable during each spline trajectory (which inadvertently causes Atlas' wrists to perform smooth human-like striking-which is exactly what is desired). The figure below and accompanying video best show this.

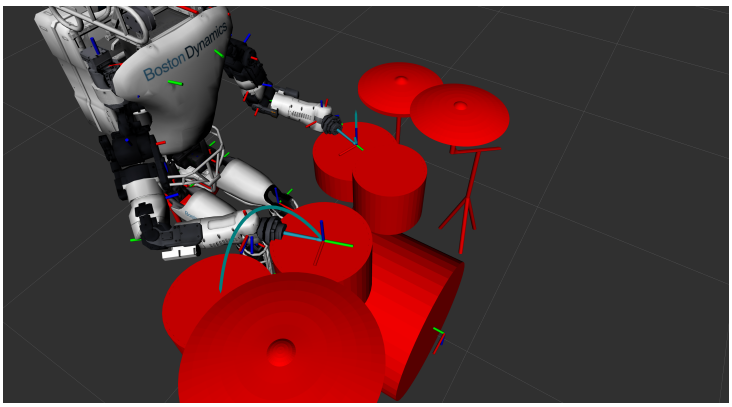


Figure 9: Drum stick parabolic trajectory.

5.2 Music Synchronization

Another feature worth mentioning is rhythmic synchronization with a song (demonstrated in video). To do this, we ensured our desired trajectories followed a desired rhythm in the segments. We chose the song "*Outstanding*" by *The Gap Band*. As described briefly in section 3.2, the song has a BPM of 98 and we encoded this in our segments to generate the trajectories.

Figure 10: *The Gap Band* album cover (Source: *MusicChartsArchive*).

Recorded and released in 1983, "*Oustanding*" was one of the major hits it's era that reached the number one spot on the *U.S. R&B Singles Chart* in February 1983 and peaked at number 51 on the *Billboard Hot 100*. We chose this song because of it's rich rhythm and classic popularity.

6 Joint Angle Analysis

Using `rosbag`, for a 25 second portion of the drumming sequence, the joint positions and velocities were plotted against time (Figure 11). The top plot shows joint positions plotted against time and the lower plot shows joint velocities plotted against time. As seen in the plot, the position plot is smooth - showing that we aren't hitting any singularities. However, in the velocity plot, there are spikes around regions where Atlas hits a drum element - these spikes literally encode a discrete action of hitting the drum and instantaneous changes in directions. Interestingly, the BPM (98 BPM in the demo) of the song that Atlas is playing along with can be recovered from these spikes by counting the number of spikes n_s in a given time interval Δt and multiplying by a factor of 60:

$$BPM = \left(\frac{n_s}{\Delta t}\right) 60$$

Worth mentioning too is that the spikes are not shooting up to large quantities. This is due to handling the singularities well with secondary tasks and the weighted pseudo inverse as described in Section 4.2. The velocity curves are differentiable with respect to the Python code simulation time step.

Figure 11: Sample plot of joint position and velocity against time.

Smooth curves are observed at approximately from $t = 4$ to $t = 7$. This is a period when Atlas has its arms resting on a drum head and dances with its torso.

We notice smooth swinging between strikes (which is what we desired). Please note however, for our this project, our main metric to measure success was realistic human-like drum playing (visually appealing). And so our main focus was on handling singularities and choosing feasible trajectories for Atlas.

A Appendix

The following image decribes the joints and linkages and was helpful throughout the project.

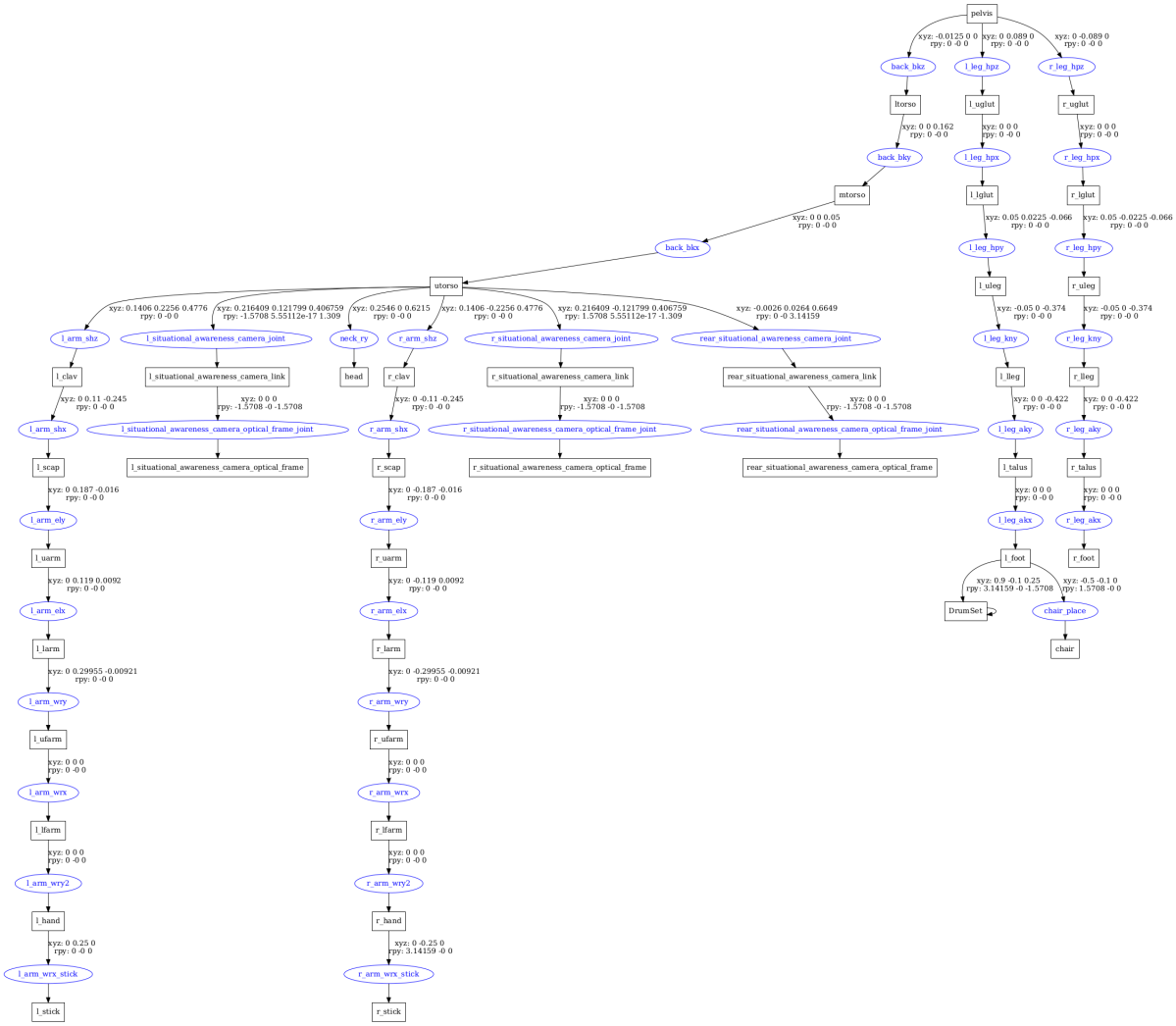


Figure 12: URDF joint and link layout.