Conun-drum: A drumming bot

Aaron Garza
Ayano Hiranaka
Abhishek Raghunathan
Paula Stocco

Spring 2022

1 Abstract

This projects consists of Toro, the robot, playing the drums in a virtual simulation environment. The environment consists of Toro seated at the center, 5 different drum set instruments, and a base. The user specifies a music score and the tempo as beats per minute using a custom designed GUI. This information is transferred to the controller using text files, which subsequently commands the bot to reach the target points at each time step. Each leg and arm of Toro are controlled independently, allowing the simulation to play up to 4 instruments at any given time. The controller is also linked to a python module that simultaneously plays the sounds of the instruments, adding authenticity in the form of sound to the simulation.

2 Introduction

There have been several attempts at programming robots to play musical instruments. Our project was inspired by Shimon, a robot that plays the Marimba (figure 1). The robot mimics user input from a keyboard and plays out the same notes on the Marimba.

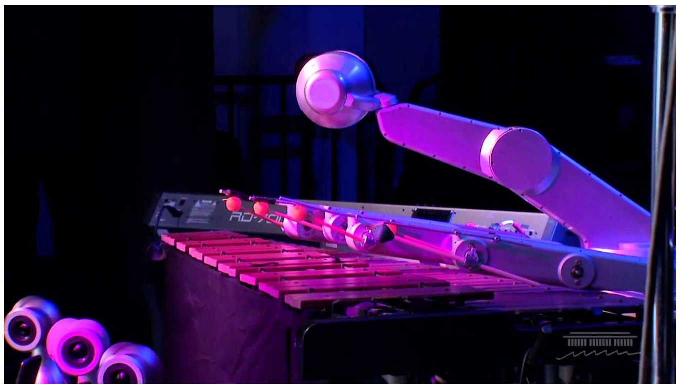


Figure 1: Shimon playing the Marimba

It should be noted that all the target points for Shimon are present in the horizontal plane. We wanted to take this one step further by controlling a bimanual robot such as Toro, to play a complete drum set, require 3D spacial manipulation and control.
The challenge for us was to be able to take user input in an artistic manner(using a GUI), efficiently transfer this information to the controller, and manipulate the 4 limbs of Toro to reach all the desired target points at the right times, while also ensuring that the sound module was synchronized and playing the right sounds at each time step. Other aspects of the project included importing a drum set and integrating it into the simulation environment, seating Toro, replacing his end effectors with drumsticks, and determining appropriate arm and leg positions, choosing target points on each drum from the CAD information,

3 Final Implementation

3.1 Environmental Modeling and System Overview

To mimic a human drummer, we select Toro, a bimanual humanoid for our simulation. Toro is equipped with a custom drumstick end effector, and is seated at a drumset placed onto a stage.
In terms of code structure, the project consists of two main modules that communicate through redis: the Sai2.0 simulation and the external Python modules enhancing the simulation. First, the user inputs a sequence of drum beats and a tempo through a GUI. The controller then receives necessasry information from the GUI to simulate Toro playing the beats inputted by the user. When Toro's drum sticks arrive at an instrument, the controller signals the audio player module to play the corresponding sound. This workflow as well as the simulation environment are illustrated in figure 2.

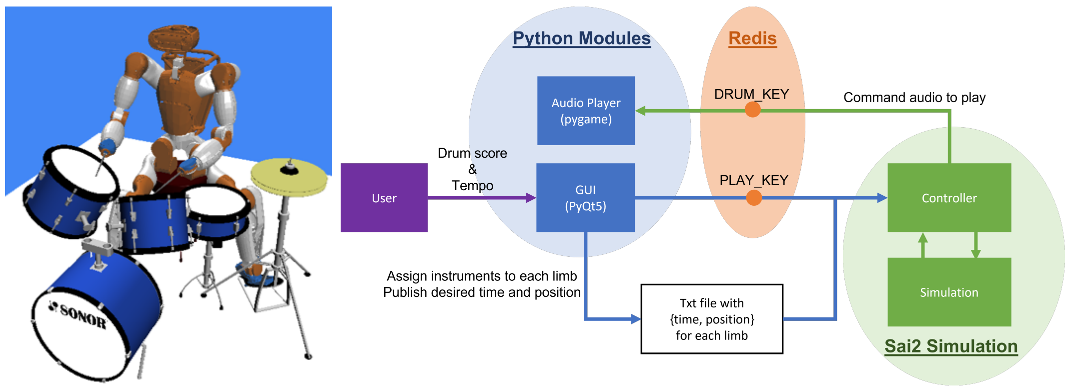


Figure 2: Simulation environment (left) and system architecture (right)

3.2 State Machines

Figure 3 below shows the high level state machine for the left and right arms of Toro. The state machine initializes into a HOME state when the user starts the GUI. In this state, the robot's joints and end effector positions are fixed in a starting configuration. Once the user has input a beat and pressed the play button in the GUI, the controller will indicate that the robot has started playing and transition to the FIRST_MOVING state. During this transition, the data from the

GUI is copied into a text file with time data corresponding to the times at which Toro will play the specified drum, and position data corresponding to the desired positions Toro's end effector must reach to play each drum. The controller then reads the data from the text files and places them into vectors that the state machine can index into. In the FIRST_MOVING state, both arms move to a starting position above the first drum they must play. For example, if the left arm's first drum to play is the snare, then the end effector will move to $(x_{snare}, y_{snare}, z_{snare}+dz)$ where dz corresponds to a fixed distance above the drum. Once both arms are in position, the arms will wait to start playing until a unified start time has been reached. In this case, we have specified that time to be 5 seconds after the play button has been pressed. This is the time at which Toro plays the first beat.

In order for the drum stick to hit the drum on time, Toro's arm must start moving ahead of time. To account for the time it takes Toro's arms to reach each drum from the ready position above the drum, the state machine factors in the velocity of the end effector when hitting, and the amount of time it might take the arm to accelerate to this velocity. The max velocity of the end effector is set to $v_{hit} = 0.5$ m/s when hitting so the $time\_to\_hit$ variable is calculated as dz/v_{hit}. The time the end effector takes to accelerate to this velocity, t_{buffer}, was estimated empirically. As an example, if Toro has to play a drum at 15 seconds, then Toro will start moving at $15 - time\_to\_hit - t_{buffer}$ seconds. Once this time has been reached, the state transitions to the HITTING state. Once the vector norm of the current position of the end effector is within 0.0001 meters to the vector norm of the desired drum position (i.e. the drum has been hit), the state machine transitions to the LIFTING_STICK state. In this state, each arm moves back up to the $z + dz$ position. When the drum stick has been lifted, the state machine will move on to the next drum that has to be hit and move to the MOVING_STATE. At this point, the state machine will move from MOVING_STATE to HITTING_DRUM to LIFTING_DRUMSTICK in a loop as Toro loops through the beat. If the stop button is pressed at any point in time and at any state in the machine, the state will transition back to HOME and Toro will move to the initial position and joint configuration. The user can input a new beat or simply press play again to let the state machine run its course.

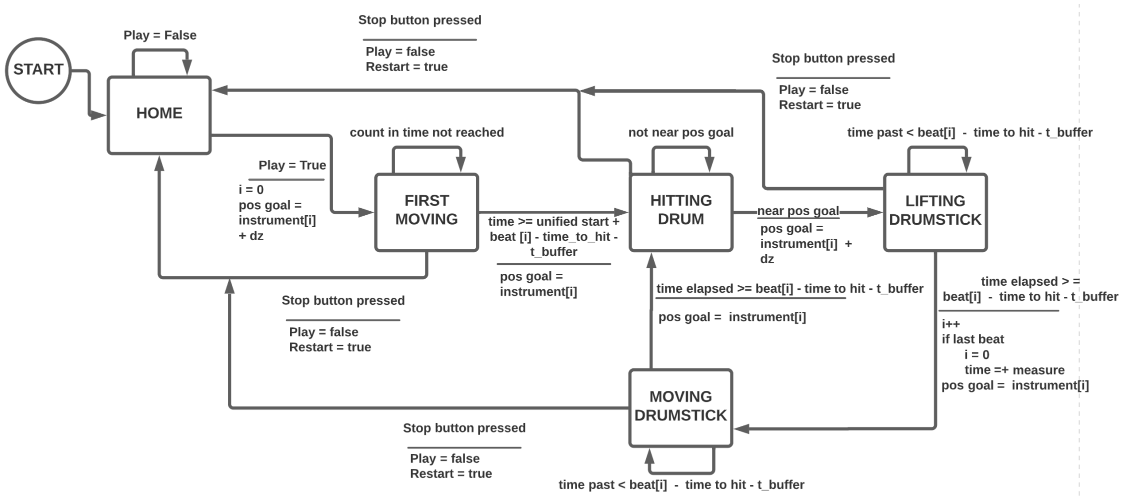


Figure 3: State machine for left and right arms of Toro

The state machine for Toro's legs shown in Figure 4 functions similarly to the state machine for the arms. The difference here is that only the joint position of the ankle is controlled since playing the bass drum and hi hat involves stomping the foot. Again, starting the GUI initializes the state machine in the HOME state and pressing play will cause a transition into the MOVING_STATE where the ankle joint moves to an initial joint position to get ready for the first stomp. Like the arms, the feet start moving ahead of time so that they finish the stomp at the right time. This calculation accounts for the angular velocity of the joints and a buffer time to account for the acceleration of the joint to the desired angular velocity. Once the time to initiate the stomp is reached, the state machine moves to the STOMP state. In this state, the machine checks if the joint angle is close to the desired angle for the stomp. After this position is reached, the state machine transitions to the WAIT state where the feet move back to their ready positions to wait for the time to initiate the next stomp. The state machine then transitions between STOMP and WAIT for the duration of Toro's playing. When the user presses the stop button, the state moves to HOME and waits until the user presses the play button again.

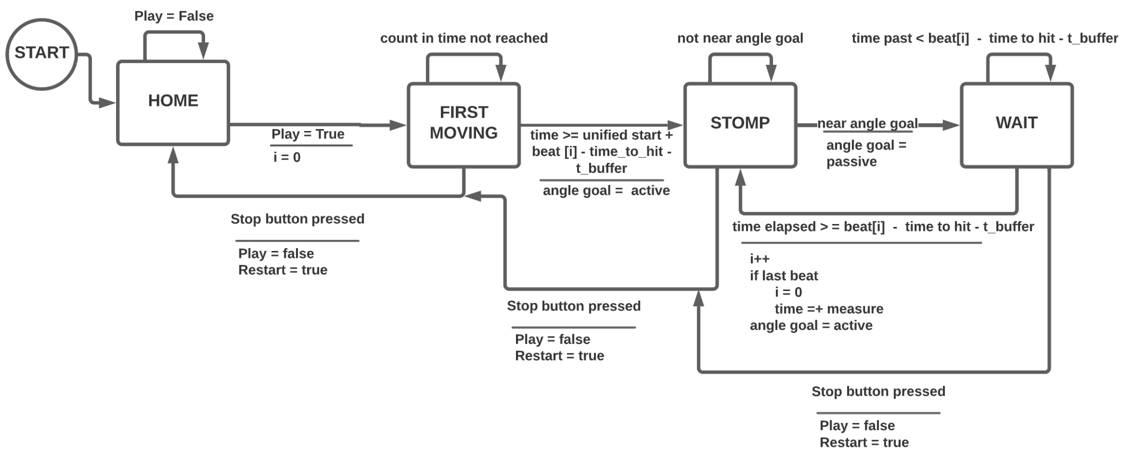


Figure 4: State machine for left and right legs of Toro

3.3 Controller

A hierarchical controller with successive Nullspaces and constrained Jacobians is used throughout the state machine. All operational space control laws are proportional derivative controllers with dynamic decoupling and trajectory tracking. Each subsequent operational space task command torques are projected onto the combined Nullspace of the previous tasks. The lowest priority law is a PD joint task. In order of highest to lowest priorities, the control laws are: hip-base position, right end effector position and orientation, left end effector position and orientation, and finally joint position. The desired joint position for left foot and right foot were dependent on the state machine, while the remaining joints were commanded to maintain their original position, preventing for example Toro's arms from rotating about his shoulders once his drum stick reached its desired position. Collisions were avoided through the state machine driven positions and trajectory planning, rather than additional repulsion fields between limbs and instruments.

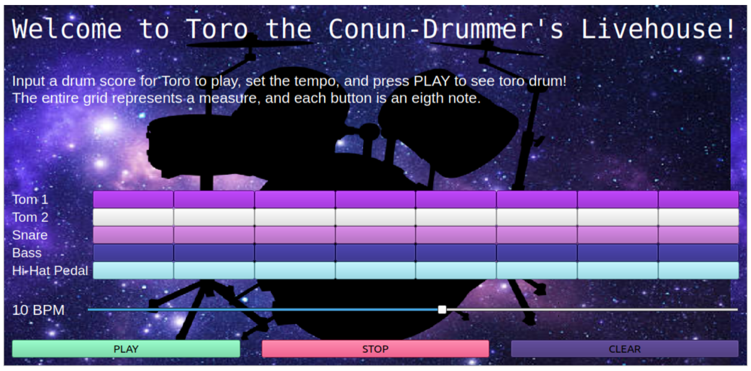
$$\Gamma = J_t^T F_t + J_{t|rh}^T F_{t|rh} + J_{t|rh|lh}^T F_{t|rh|lh} + J_{t|rh|lh|j}^T F_{t|rh|lh|j}$$

where
Γ is the total commanded torques
J is a constraint-consistent and task-consistent posture Jacobian
F is the calculated forces including dynamic decoupling in the operational or joint spaces

This hierarchical controller with dynamic decoupling was chosen to isolate the limb tasks and reduce motion coupling. Velocity saturation was initially used, under the hypothesis that achieving a maximum velocity as quickly as possible would produce the most repeatable downward motion from lifted drumstick to hit position. However, this method stops applying motor torques once velocity is reached and thus performed less reliably than trajectory tracking which computes intermediate desired positions using our specified maximum velocity, acceleration, and jerk, and will actively attempt to correct for position error during travel.

3.4 User Interface

A graphical user interface (GUI), shown in figure 5, is developed using PyQt5 library. The grid of toggle buttons located in the center of the window represents a measure of a drum score, and each column represents an eighth note. The time signature is common type. The user can click buttons in the grid to select which instrument Toro will play at each beat, and set a tempo using the tempo slider. Clicking the CLEAR button clears the user's beat selection. When the PLAY button is clicked, the GUI first confirms that the user input is valid (i.e. the input beat does not require more than two arms), and displays an error message if the input is impossible. If the input is valid, each instrument is assigned to each of Toro's four limbs in a way that prevent Toro's arms from crossing each other. Then, a text file containing desired end effector position and time pairs for a measure are exported for each limb. This file is read by the controller. Finally, a redis flag is set signaling the controller that the data preparation step is complete. At any point while the simulation is running, the user can click the STOP button, which sets a redis flag signaling the controller to reset, putting the simulation in a state allowing new user input.


Welcome to Toro the Conun-Drummer's Livehouse!

Input a drum score for Toro to play, set the tempo, and press PLAY to see toro drum!
The entire grid represents a measure, and each button is an eigth note.

Figure 5: Graphical user interface accepting drum score and tempo from users

3.5 Sound Generation

An important aspect of this project was to include drum sounds to accompany Toro's playing. A separate Python module using the Pygame library's audio capabilities was used for playback of downloaded drum samples. Since Toro can play multiple drums at once, Pygame's mixer module was perfect for our application since it allows for multiple sounds to be played without incorporating blocking code or obstructing sounds that have already started playing. The Redis server communicates between the controller module and sound module. Separate keys for each drum were made to indicate whenever a drum's corresponding sound should play. In the controller, whenever Toro has played a drum, the appropriate drum Redis key is set to 1. The Python sound module constantly checks the values of these keys, plays the appropriate sound whenever a key is read as 1, and sets the value of the key back to 0.

3.6 Results

We found a 0.02 second difference between instrument sounds is audibly detectable. The final controller is able to play most configurations with error between any of the four limbs consistently below 0.02 seconds. The following analysis is an illustration of the team's challenge in perfecting consistent timing, and is not intended as a deep investigation into the cause, which is contained in the challenges section. The musical measure shown in Table 1 was set through the GUI and the data visualized as an example of the entire systems behavior:

Instrument	Beat
Tom2	2, 4, 6, 7, 8
Tom1	3, 5, 6, 8
Snare	1, 5, 7
Bass	3, 4, 5, 6, 7, 8
Hi-hat	1, 2, 5, 6, 7, 8

Table 1: Instruments playing each of eight beats, repeated after each measure

Figure 6 shows the largest difference between all the limbs for a repeated eight beat measure. For this note combination, every fifth beat when both arms and legs are playing is more than the audible delay between notes.

If the hit times for the instruments playing during the note of interest, beat 5, are reviewed using Figure 7, it can be seen that it is not a general delay but rather an early foot hits causing the large difference between instrument hit times.

Additionally, during other musical combinations not pictured, one limb may be off audibly but not consistently. There are multiple factors affecting note playing, which are discussed further in the challenges section along with suggested improvements.

3.7 Discussion

One of the team's goals was to have Toro play at 15 BPM, and not only appear visually on time but have a sound component that truly reflected the controller capabilities. As described in the Final Implementation section, one of the challenges was tuning the controller such that all limbs always played within the non-distinguishable 0.02s window from one another. This task was made harder at higher speeds.

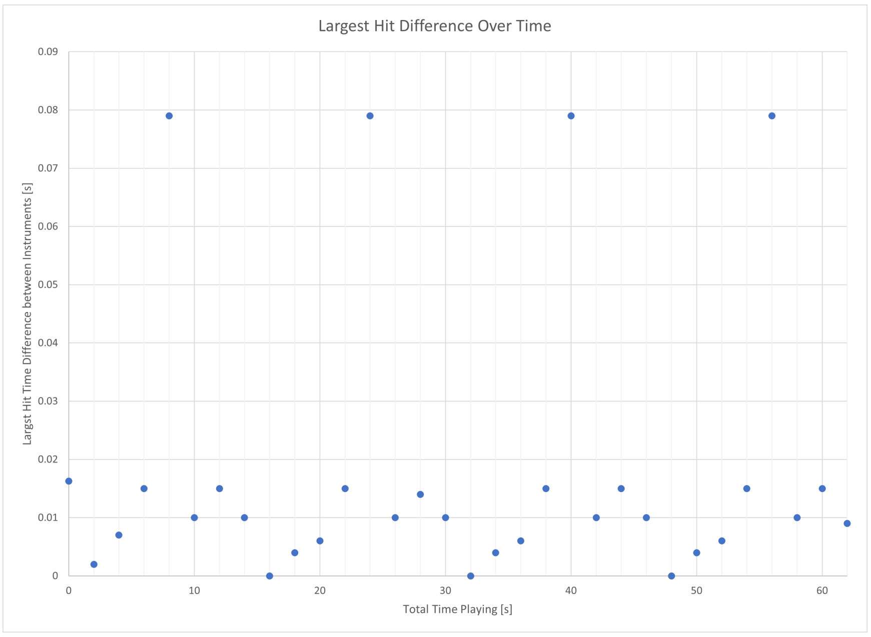


Figure 6: Largest difference between any two instruments at every beat

During tuning, we found that print statements and redis key reads in occasionally accessed portions of the code could affect processing time and thus controller performance. While we attempted to optimize the code, this effect may still account for some notes being off beat.

The current controller design could contribute to the challenge of tuning Toro, because the nature of the hierarchical control prevents lower priority tasks from affecting higher priority ones but not vice versa.

To illustrate,the closed loop system equations to tune the PD parameters of the right hand are shown below. Gravity, centrifugal, and Coriolis affects are left out of the following analysis.

The dynamics are written using the operational space inertia matrix and operation space coordinates for the right hand end effector, and the dynamically consistent Jacobian inverse used to map the control torques into the appropriate space.

$$\hat{\Lambda}_{rh}\ddot{x}_{rh} = \bar{J}_{t|rh}\Gamma$$

The left hand end effector and joint tasks are only in the Nullspace of the right hand end effector task therefore dropping away, and being in its own space the right hand end effector control force is left.

$$\hat{\Lambda}_{rh}\ddot{x}_{rh} = \bar{J}_{t|rh}J_t^T F_t + F_{t|rh}$$

The PD constants to be tuned are within the right hand force equation. However, regardless of dynamic decoupling and any chosen k_v and k_p values, the affect of the hip control task $\bar{J}_{t|rh}(J_t^T F_t)$ will cause an unaccounted for disturbance. If the hip-base position task happens to fall within the right arm task space, it can cause an error in position.

Methods we would consider if stating anew for addressing this are included in the challenges section.

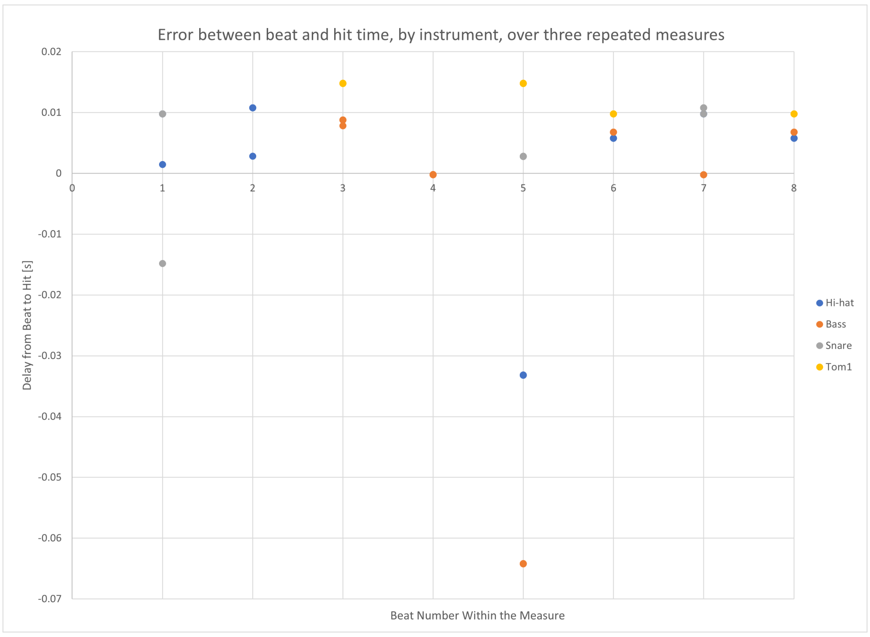


Figure 7: Largest difference at every beat for all instruments

4 Challenges

The Conon-drum project gave us the opportunity to explore multi-limb control with the Toro robot. Working with Toro posed many challenges which we reflected on to consider how similar problems might best be addressed in the future.

4.1 Faster BPM and Consistent Timing

We found that differences in controller and simulation loop rate made a large difference; team members who could not run at 1 kHz often encountered repeated drum hits or large delays in hits that were never seen at 1 kHz run time. Similarly, we found that print statement and Redis key reads would affect the processing time and therefore the hit time. In addition to optimizing run time, several improvements could be made to the controller to address current weaknesses that posed a challenge.

Trajectory tracking was implemented for both arm tasks, and found to be fairly consistent. However, an error in the start position, which was set as a given vertical displacement from the drum hit location, would generate different trajectories and therefore different hit times. To remove steady state error from the initial position and integral term could be added to the current PD controller.

A difficulty in producing consistent timing at faster tempos in particular was the non-instant reaction of the hand position reaching its desired pose. Although the threshold distance for reaching desired position was reached and registered, Toro was unable to stop or reverse direction quickly enough with an enforced acceleration and jerk. One way to address this could be to implement a collision between the end effector and the drum. The hit would force Toro to stop and return immediately; inconsistent trajectories upwards caused by impact would be acceptable because time to reach goal position for states other than hitting drum does not affect instrument hit time as long as the wait position is reached in time. See 8 in the Appendix for a screen shot showing a preliminary implementation of this idea on the Conun-drum bot.

The challenge of higher priority tasks affecting lower ones could be addressed with a feed forward term added to all non-first priority tasks.

Another method to address this effect would be to combine all tasks into a single Jacobian, which could make the error between them smaller as one task would not be enforced above another. Compared to the equations shared in the controller section, the PD constant equations would be solved simultaneous to achieve a critically damped system.

Additionally, centrifugal and Coriolis forces were not compensated for in our controller, and doing so could also improve its accuracy.

4.2 Head Nod

Early in the project, nodding Toro's head to the beat was incorporated into the joint task to help visualize the count in time. However, when fine tuning the hit times we discovered this task affected the final controller accuracy and it was removed. This could be remediated using a hierarchical joint controller, see 9 in the Appendix for preliminary code implementation. Alternatively, the aforementioned (subsection 4.1) combined task controller could be used with the head task subsequently projected into the null space.

5 Conclusion

Playing the drums like a human is certainly an unusual task for a robot. While we often view robots as accurate, tireless, and lifeless machines, our experience working with "Toro the Conun-Drummer" contradicted with this norm. Customizing Toro's drum sets, implementing, improving, and tuning our controller, and watching Toro get better at drumming was a heartwarming experience - as if we were watching a child learning something new. As robots take on more and more roles in society, it is not only important for them to be safe and reliable, but also likeable to humans. We believe robots like "Toro the Conun-Drummer" challenging itself with atypical, human-like tasks can help change the impression of robots as a human's "tool" to a "skilled collaborator." In our attempt to mimic a human drummer, we were also greatly impressed by the human drummer's skills, consistently hitting the instruments at millisecond-scale accuracy. Although our "Conun-Drummer" is not skilled enough to jam with human musicians yet, we hope to see robots joining humans in creative tasks in the near future.

Visit https://www.youtube.com/watch?v=IfrgMBxHBiM to see Toro in action.

6 References

Free CAD files used for drum set:
https://grabcad.com/library/drum-set-1/details?folder$_i d = 1505224$

Free Bass Drum sound effects:
https://www.fesliyanstudios.com/royalty-free-sound-effects-download/bass-drum-274

7 Appendix

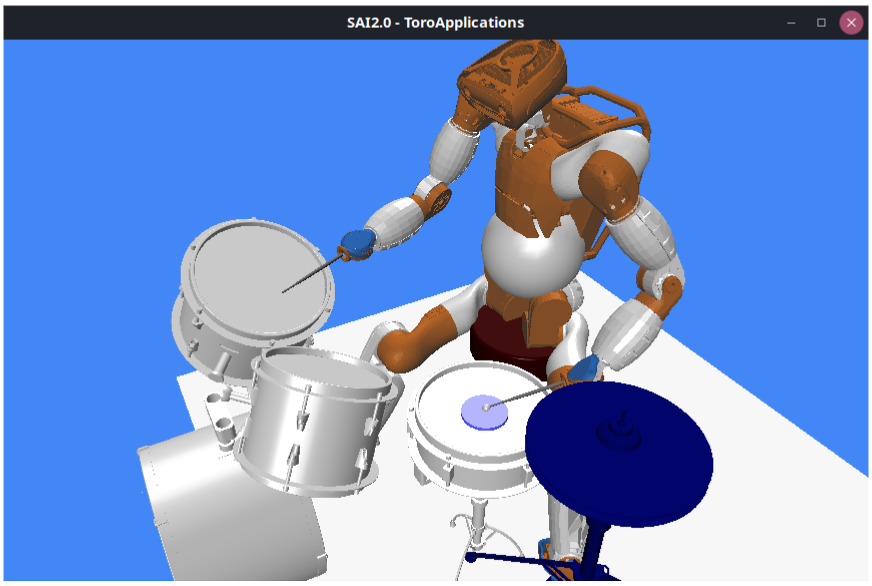
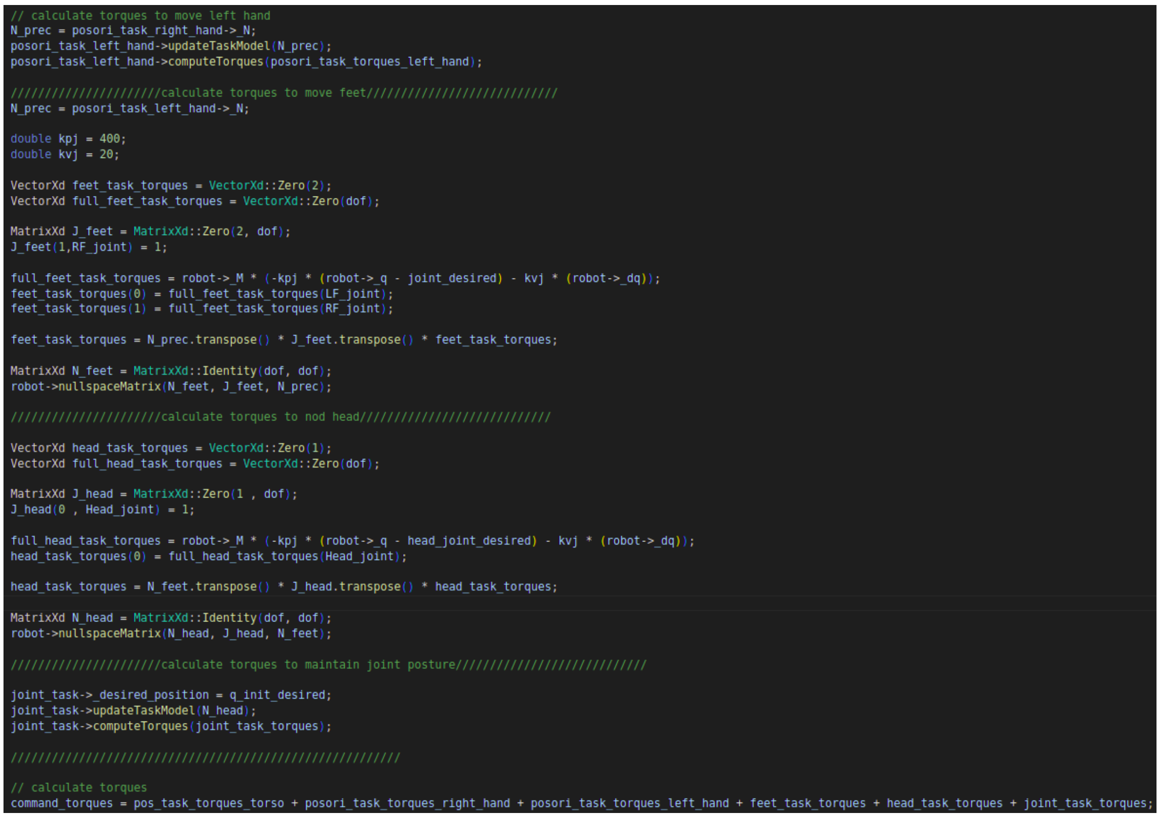


Figure 8: Collision objects on the drum and Toro's hand were added and collision checking incorporated.



```cpp
// calculate torques to move left hand
N_prec = posori_task_right_hand->_N;
posori_task_left_hand->updateTaskModel(N_prec);
posori_task_left_hand->computeTorques(posori_task_torques_left_hand);

//////////////////////calculate torques to move feet///////////////////////////
N_prec = posori_task_left_hand->_N;

double kpj = 400;
double kvj = 20;

VectorXd feet_task_torques = VectorXd::Zero(2);
VectorXd full_feet_task_torques = VectorXd::Zero(dof);

MatrixXd J_feet = MatrixXd::Zero(2, dof);
J_feet(1,RF_joint) = 1;

full_feet_task_torques = robot->_M * (-kpj * (robot->_q - joint_desired) - kvj * (robot->_dq));
feet_task_torques(0) = full_feet_task_torques(LF_joint);
feet_task_torques(1) = full_feet_task_torques(RF_joint);

feet_task_torques = N_prec.transpose() * J_feet.transpose() * feet_task_torques;

MatrixXd N_feet = MatrixXd::Identity(dof, dof);
robot->nullspaceMatrix(N_feet, J_feet, N_prec);

//////////////////////calculate torques to nod head///////////////////////////

VectorXd head_task_torques = VectorXd::Zero(1);
VectorXd full_head_task_torques = VectorXd::Zero(dof);

MatrixXd J_head = MatrixXd::Zero(1 , dof);
J_head(0 , Head_joint) = 1;

full_head_task_torques = robot->_M * (-kpj * (robot->_q - head_joint_desired) - kvj * (robot->_dq));
head_task_torques(0) = full_head_task_torques(Head_joint);

head_task_torques = N_feet.transpose() * J_head.transpose() * head_task_torques;

MatrixXd N_head = MatrixXd::Identity(dof, dof);
robot->nullspaceMatrix(N_head, J_head, N_feet);

//////////////////////calculate torques to maintain joint posture///////////////////////////

joint_task->_desired_position = q_init_desired;
joint_task->updateTaskModel(N_head);
joint_task->computeTorques(joint_task_torques);

/////////////////////////////////////////////////////////////

// calculate torques
command_torques = pos_task_torques_torso + posori_task_torques_right_hand + posori_task_torques_left_hand + feet_task_torques + head_task_torques + joint_task_torques;
```

Figure 9: Code to create a hierarchical joint controller keeping Toro's head nod in the Nullspace