Human and Exoskeleton Interaction Simulation

Charles Bales, Josephine Bowen, Shreyas Chandra Sekhar, Michael Keable, and Tanuj Sane
RBE 580 Biomedical Robotics

***Abstract*—In this project the team tested and validated the effects of human joint forces on a lower-body exoskeleton device using a human model and exoskeleton model implemented in the Asynchronous Multi-Body Framework (AMBF). The two models were secured together with rigid body straps at several points to simulate the real-world supports. Torques were applied to the human model joints using AMBF-specific ROS clients, with similar clients used to measure the resulting forces on the exoskeleton joints. Utilizing acceleration kinematics, the exoskeleton forces were calculated as validation for these client outputs. The torques and forces on the joints were also analyzed with these client outputs.**

***Index Terms*—AMBF, exoskeleton**

I. Background

Exoskeletons are wearable devices that are placed on the body of an individual to provide added support and improve the user's strength [1]. The three main industries that use exoskeletons are healthcare, construction, and military [2]. They are most often used in the healthcare industry to help paraplegics and amputees walk again. Exoskeletons can also be used to help patients who have suffered a stroke or spinal cord injury to regain everyday functions. In the construction industry, exoskeletons help to support the wearer during heavy lifting to reduce the chance of back injuries. Military exoskeletons may be used to help injured or wounded soldiers, or to augment the capacity of the wearer. They allow soldiers to carry up to 17 times more weight than normal [2].

An exoskeleton can be as simple as a passive rigid supportive device, or as complex as an artificially intelligent machine. Sensors and actuators can also be added to the exoskeleton to increase functional potential [3]. One specific area of focus is soft robotic exoskeletons, which ideally reduce the chances of injury. Additionally, they are easier to fit to the patient, and improve the maneuverability of the patient. Exoskeletons may be affixed to the patient or may be separate mobile devices.

Control of exoskeletons is important in keeping the user safe and providing the best assistance possible. New control learning methods are being developed such as the ones detailed by Huang et al. and Zhang et al.

Huang et al. states that the exoskeleton controller should cause little interaction forces between the human and the exoskeleton. This can be done by either sensor-based or model-based controllers. For sensor-based controllers output torques are calculated leading to the exoskeleton to move, giving senor-based controllers the ability to varying motion [4]. Model-based controllers reduce the complexity of sensory systems, but do not handle varying motion well [4]. Huang et al. detail their proposed Hierarchical Interactive Learning (HIL) control strategy taking from both the senor and model based controllers [4].

Zhang et al. investigate the zero-moment point (ZMP) stability margin to reduce the interference between the human and the exoskeleton in order to maintain stability during walking [5]. They conclude that if the ZMP is at the same point as the support area, significant interference occurs between the human and the exoskeleton. To minimize the interference new controls were developed based on the human exoskeleton interaction forces [5].

The research presented in this paper experiments with a coupled human-exoskeleton model. A custom human body model is coupled with an existing exoskeleton model in simulation to determine the forces and torques (collectively, torques) induced on the exoskeleton as a result of torques induced on the human body. A software module is developed which estimates joint torques on the exoskeleton given the experimental data, for use in an exoskeleton control stack.

II. Methodology

The coupled human-exoskeleton model is simulated in the Asynchronous Multi-Body Framework (AMBF) [6]. The exoskeleton model was created during previous research. The human model was custom-generated for this application. Strap meshes for securing the human model legs to the exoskeleton were created. Joint torque and force calculations were organized to validate

A. Human Model

The human model was generated using Blender [7], a 3D modeling and rendering software. The body parts included the torso, the head, and the left and right thighs, shanks, and feet. They were designed in Solidworks and were exported as STL files, which were then imported into Blender, as seen in Fig. 1. Each part was sized to fit into the exoskeleton once completely assembled.

Once there, an AMBF [6] plugin developed for Blender was used to assemble the human model parts. This plugin allows for streamlined placement of limb joints and specification of physical parameters such as mass and friction values. Joints simulating the hips, knees, and ankles were placed into the model. Each leg was modeled as a planar mechanism consisting of a thigh and a calf. Joint limits were imposed on the hip, knee, and ankle, based on the restrictions already imposed on the exoskeleton's movement.

Once the physical parameters were set, the plugin was able to generate an ADF file containing the pertinent information necessary for AMBF to load in the human model. The model

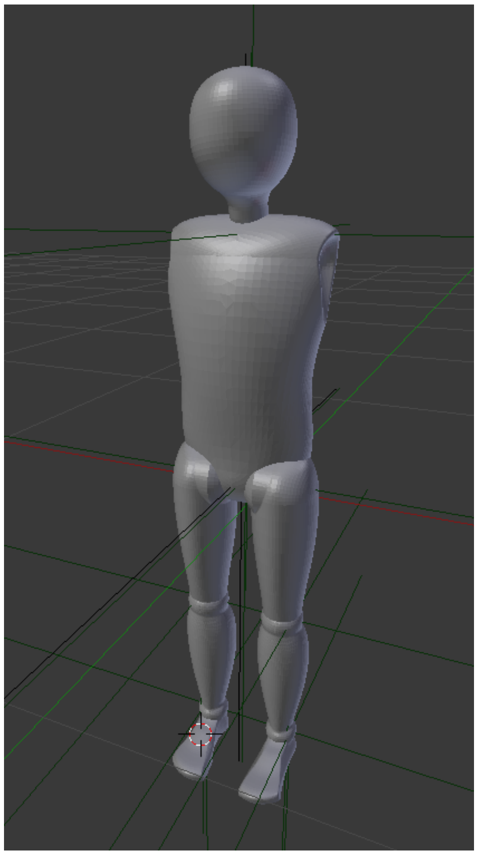
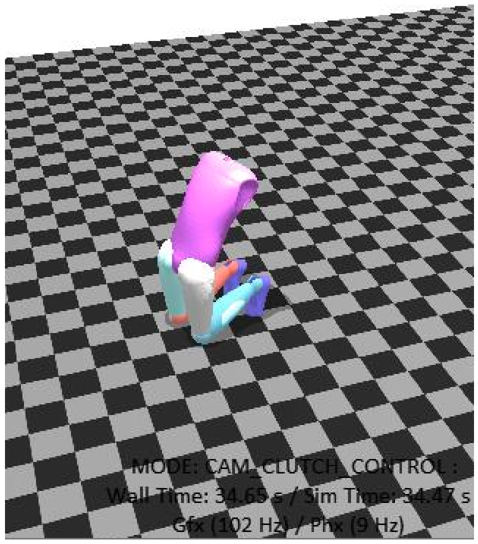


Fig. 1. Human model in Blender

TABLE I
HUMAN MODEL JOINT LIMITS

Joint	Min	Max
Hip	$-30°$	$110°$
Knee	$-110°$	$0°$
Ankle	$-30°$	$45°$



MODE: CAM_CLUTCH_CONTROL :
Wall Time: 34.65 s / Sim Timer: 34.47 s
Gfx (102 Hz) / Phx (9 Hz)

Fig. 2. Human model in AMBF

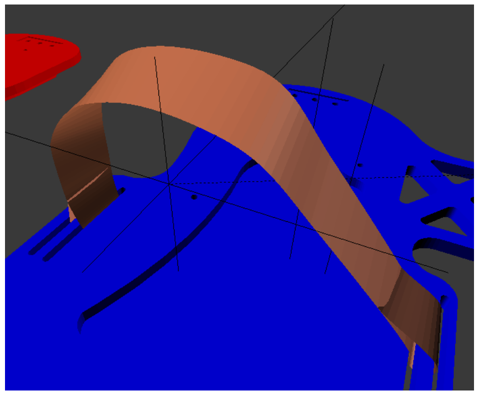


Fig. 3. Exoskeleton straps in Blender

was tested and was successfully loaded as seen in Fig. 2, meaning it was ready for integration with the existing exoskeleton model.

B. Exoskeleton Straps

Straps were implemented to connect the exoskeleton model to the designed human model. Using Blender, the exoskeleton was uploaded and the holes in each limb section that exist to hold the straps were used as a reference point for drawing curve geometries. These curves were then extended over the contour of the rest of the relevant leg section and where turned into meshes, as seen in Fig. 3. The meshes required cleaning to ensure that they did not overlap with sections of the human model but were otherwise appropriately sized.

C. Integration

The human and exoskeleton models were both designed in Blender and tested in AMBF. The human model was then placed into the straps and secured, as seen in Fig. 4.

Due to collision issues in the AMBF implementation, the straps had to be treated as fixed mounting points for their connection to the human leg. This is opposed to the treatment of straps as independent bodies that keep limbs attached via encirclement. While not as accurate to a physical system, the fixed connection was still similar enough in function to a tightly secured strap.

One issue that had to be resolved was that since the hip, knee, and ankle joints of the human model were at different vertical points than the corresponding exoskeleton joints, securing each limb section to each strap would prevent the leg from bending appropriately. To amend this, connections between leg and strap for the thigh and shank sections were removed, leaving the foot and body/hip connections in place. This allowed for the legs to bend while keeping the feet and torso attached to the appropriate mounting points.

Once the two models were secured in Blender, the ADF file for AMBF had to be generated correctly. The assembly was tested multiple times in AMBF with a few issues arising, such

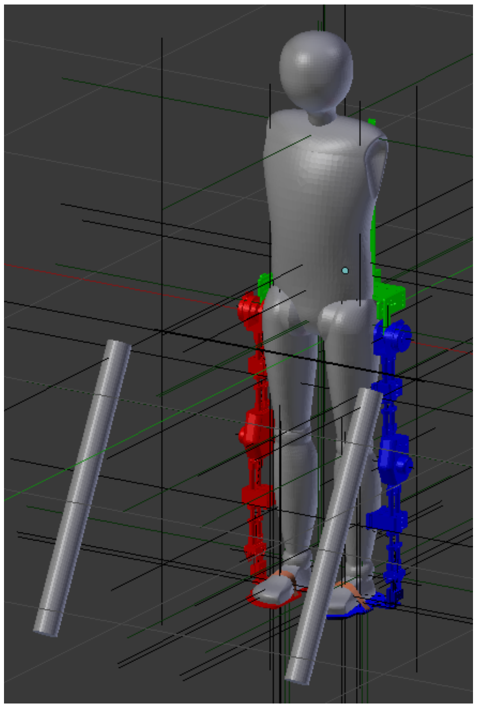


Fig. 4. Human and Exoskeleton assembly in Blender

as the aforementioned joint lock and odd collision groupings that would cause the human and exoskeleton to helicopter into the air at high speeds. These were amended, and the final iteration of the Blender assembly was able to kneel with support from the crutches, with the human remaining secured within the exoskeleton.

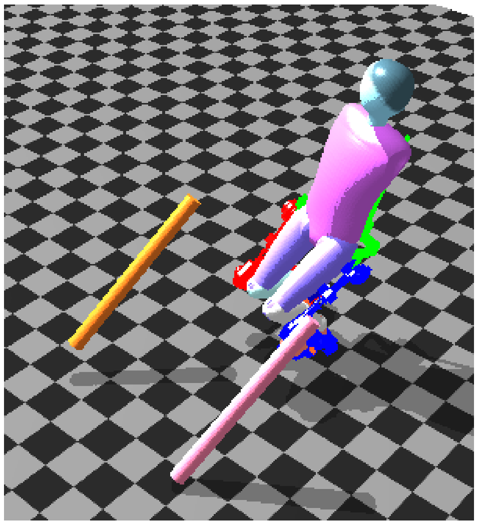


Fig. 5. Human and Exoskeleton assembly in AMBF

D. Validation

To ensure that the received data was correct, validation was performed mathematically by formulating the kinematics of the model. The expected output from the validation script was the instantaneous applied forces on the foot.

The statics of the human body model are used to validate the joint torques on the human body model in response to an applied external force at the hip. The legs are modeled as planar serial manipulators with the base at the hip of the human body model. This simplifies the formulation of the kinematics. The coordinate frames at each joint are assigned according to Denavit-Hartenberg frame constraints 6.

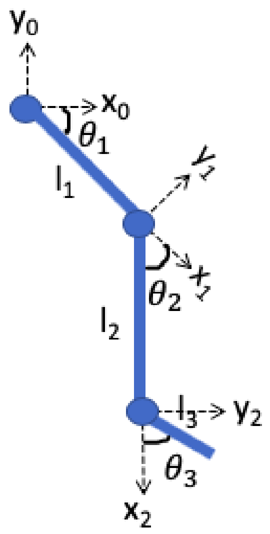


Fig. 6. Human leg with DH-parameter frame assignments

The joint torques of a serial manipulator are related to the force applied by the end-effector (here, tip force) by the following equation.

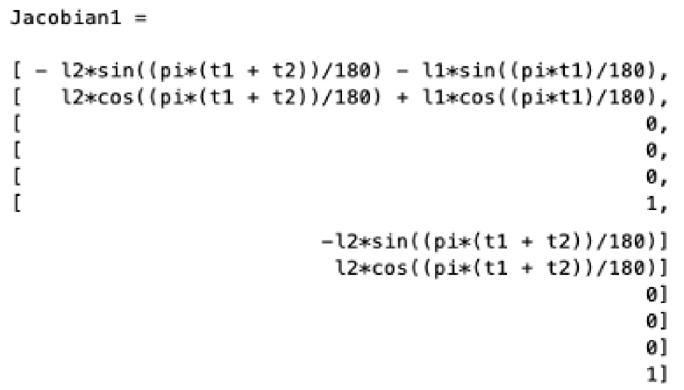
$$\tau = J(\vec{q})^T \cdot \vec{F_{tip}} \tag{1}$$

Here, $J(\vec{q})$ is the Jacobian of the manipulator, evaluated when the manipulator is in configuration $\vec{q}$. The Jacobian of the legs of the human model is found via the velocity kinematics, and is displayed symbolically below as a function of the joint angles tn for each of the n joints.

1 can be used to find the vector of joint forces given the found Jacobian and an input torque vector at the hip. This vector, shown symbolically below in figure 8, should match the external force output given the measured joint torque values. The link lengths of the upper and lower leg were inputted into the software. The theta values were inputted as zeros as the torque is being applied to model with a straight leg. A torque value of 50 N/m in the x direction was inputted into the software and the value for Ftip was calculated in variable form. The solved Jacobian matrix can be seen in figure 9 and the solved Ftip vector can be seen in figure 10.

E. Experimental Setup

The combined exoskeleton-human model was imported into AMBF, lifted off the ground plane and left free hanging. A

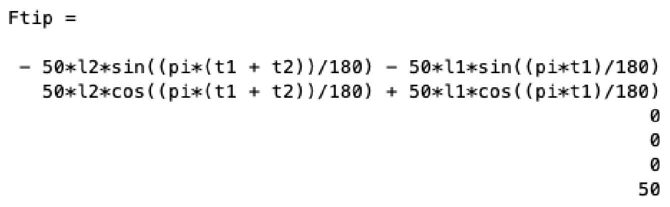


```
Jacobian1 =

[ - l2*sin((pi*(t1 + t2))/180) - l1*sin((pi*t1)/180),
[   l2*cos((pi*(t1 + t2))/180) + l1*cos((pi*t1)/180),
[                                                   0,
[                                                   0,
[                                                   0,
[                                                   1,

                              -l2*sin((pi*(t1 + t2))/180)]
                               l2*cos((pi*(t1 + t2))/180)]
                                                        0]
                                                        0]
                                                        0]
                                                        1]
```

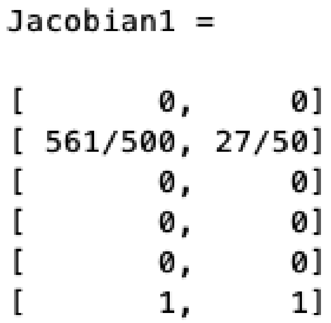
Fig. 7. Symbolic manipulator Jacobian



```
Ftip =

 - 50*l2*sin((pi*(t1 + t2))/180) - 50*l1*sin((pi*t1)/180)
   50*l2*cos((pi*(t1 + t2))/180) + 50*l1*cos((pi*t1)/180)
                                                         0
                                                         0
                                                         0
                                                        50
```

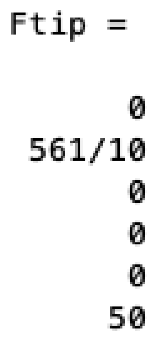
Fig. 8. Symbolic force vector



```
Jacobian1 =

[       0,     0]
[ 561/500, 27/50]
[       0,     0]
[       0,     0]
[       0,     0]
[       1,     1]
```

Fig. 9. Numeric Jacobian matrix



```
Ftip =

       0
  561/10
       0
       0
       0
      50
```

Fig. 10. Numeric force vector

parallel model of the system was also generated using RBDL, a library that computes dynamics of rigid bodies. The model in AMBF served as the ground truth of the system, while the RBDL model was used only for inverse dynamics calculations.

The system was commanded to follow a trajectory. During each timestep, the desired instantaneous joint velocity and joint acceleration values were evaluated from the trajectory and fed into a proportional-derivative (PD) controller responsible for setting the joint velocity and acceleration to the given values. The actual values of the joints were read back from the simulated model in AMBF and the error in joint position, velocity, and acceleration were computed.

These errors were used in calculation of the required joint torques to bring the robot back onto the desired trajectory, via inverse dynamics. This calculation was carried out using the RBDL model of the system.

The above process is repeated until completion of the trajectory. The resulting motion was to follow the trajectory and indicate that any deviation was handled by an increase in joint torques. The entire process can be visualized as follows in Fig. 11.

III. Results and Conclusion

A. RBDL

RBDL was used as a way to test the joint torque and force interactions of the assembled model in AMBF. A desired trajectory was created for applying forces on the thigh joints to simulate a walking motion and inverse dynamics were performed to calculate a desired set of joint angles and the driving torques necessary for performing this motion. The actual joint angles and torques of the model experiment were then measured and compared to the desired values. The error between these sets was noted. Finding this error allowed us to determine reasons for motion discrepancies and also allowed us to implement a PD controller for better matching this desired motion. At this time, torque values are similar, but the results are not always consistent. This is likely due to a need to further tune our PD gains and because of small differences in parameters between the RBDL implementation of the model and the AMBF model itself. Two graphs demonstrating the joint torque values of the thighs being tested with RBDL can be seen in Fig. 12 and Fig. 13. Fig. 14 demonstrates issues with the PD controller in regulating joint velocities.

B. Validation

Once force outputs are generated from RBDL the validation script that was generated can easily be used to validate the results of the software. The script that was written takes the applied torque values and multiplies them by the calculated Jacobian matrix to get the resulting force values. Currently the program takes in an input torque of 50 N/m in the x direction and solved for the resultant force vector. However, the program can easily be adapted to whatever torque is being applied to the model in RBDL and the output force will be able to be compared to the force that is being calculated by the software.

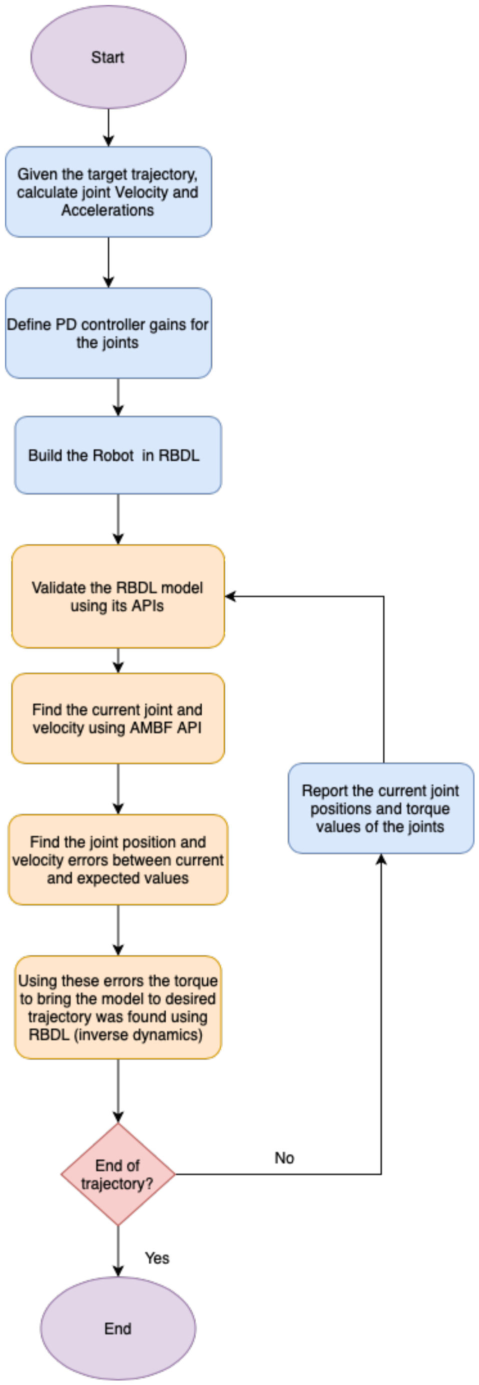

Start
Given the target trajectory, calculate joint Velocity and Accelerations
Define PD controller gains for the joints
Build the Robot in RBDL
Validate the RBDL model using its APIs
Find the current joint and velocity using AMBF API
Find the joint position and velocity errors between current and expected values
Using these errors the torque to bring the model to desired trajectory was found using RBDL (inverse dynamics)
End of trajectory?
No
Yes
End
Report the current joint positions and torque values of the joints

Fig. 11. Code Work Flow

IV. Discussion

A. Implementation

The torques and positions of each joint were found from the simulation in AMBF. This was done by applying a set torque value to the left and right thighs and using inverse dynamics to solve for the torque and position values at each of the joints. However, since the motion is not always perfect the torque values are not consistent between the human and

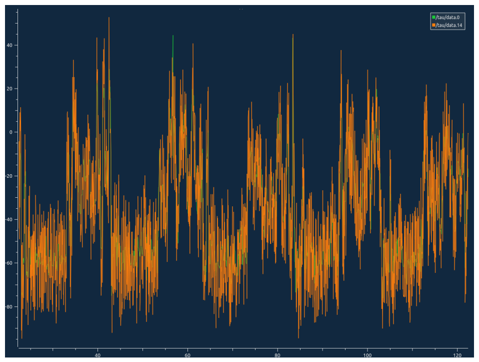


Fig. 12. Left thigh torque

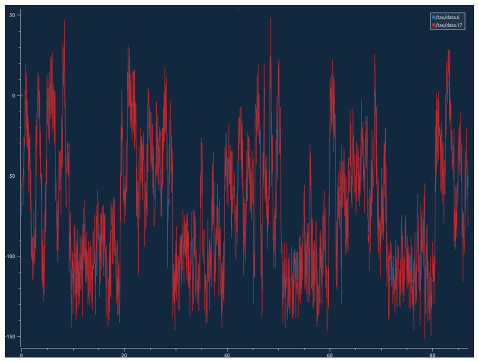


Fig. 13. Right thigh torque

exoskeleton model. Due to inconsistency in torque values the validation program was unable to be used to compare the results. The validation program was tested with a 50 N/m torque in the x direction but the resultant force vector was not able to be compared to the results from RBDL due to these inconsistencies. The validation is set up to take in any torque value; however, further updates still have to be made to the RBDL program before in can be validated using the MATLAB code.

B. Contributions

Listed here are the key parts of this project and the team members that contributed to each.

- Paper writing (All)
  - Adding explanations, discussions, and technical contributions to each paper submission.
- Presentation assembly (All)
  - Putting together project presentations and recordings
- Blender models (Charles, Shreyas)
  - Human model and strap construction, exo-human model assembly
- Validation calculations (Josephine)
  - MATLAB system for validating RBDL code force and torque outputs
  - Testing system with theoretical applied torque value
- AMBF implementation and RBDL force and torque testing (Shreyas, Charles)

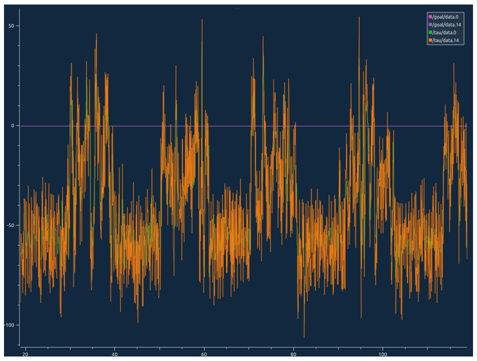


Fig. 14. Joint velocities

  – Loading human and exoskeleton models into AMBF and using RBDL to test force and torque inputs on joints.

The link to a video summary of this project can be found here: https://youtu.be/gFaPmAqWJc4

V. Further Work

The team determined multiple areas of future work for this project. The first would be to determine the reasoning as to why the torque values are going out of sync. When the software is run, the majority of the time, the torque values are in sync. However, the values are not always consistent and the reasoning behind this is currently unknown. Another area of future research would be to further tune the PD controller. A third area of future research would be to add guided motion to the model so that the model can actually complete a walking motion. Once the guided motion is added to the model then the torques and forces and be determined during the walking motion. The final area of future research would be to build upon the current validation methods. A more in depth calculation into inverse dynamics could also be added to the validation. Finally, the issue mentioned in the methodology discussion regarding the straps causing unexpected collision effects can be looked into. Further streamlining the collision geometries of the human and strap meshes should solve this problem.

References

[1] A. F. Ruiz-Olaya, A. Lopez-Delis, and A. F. da Rocha, "Chapter eight - upper and lower extremity exoskeletons," *Handbook of Biomechatronics*, pp. 283–317, 2019.

[2] P. van der Schaft. Exoskeletons strengthen in uses beyond healthcare. [Online]. Available: https://www.roboticsbusinessreview.com/robo-dev/exoskeletons-uses-beyond-healthcare/

[3] B. Marinov. What is an exoskeleton? [Online]. Available: https://exoskeletonreport.com/what-is-an-exoskeleton/

[4] H. Rui, C. Hong, G. Hongliang, L. Xichuan, and Z. Jianwei, "Hierarchical learning control with physical human-exoskeleton interaction," *Information Sciences*, pp. 584–595, 2018. [Online]. Available: https://doi.org/10.1016/j.ins.2017.09.068

[5] X. Zhang, G. Wang, P. Yuan, H. Xu, and Y. Hou, "A control strategy for maintaining gait stability and reducing body-exoskeleton interference force in load-carrying exoskeleton," *Journal of Intelligent & Robotic Systems*, vol. 97, pp. 287—-298, 2019.

[6] W. Y. G. R. . G. F. Munawar, Adnan, "A real-time dynamic simulator and an associated front-end representation format for simulating complex robots and environments," IROS, IEEE, 2019, Dec, 2019.

[7] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org