Wearable Robotic Third Arm 2021 Fall Semester Report

Designing an Online Controller for the Wearable Robotic Third Arm for Self-Handovers with Collision Avoidance

Raphael, A, F, Fortuna

Cornell University, raf269@cornell.edu

The Wearable Robotic Third Arm previously used an offline controller to go to a predetermined location and deliver an object to the other hand of the human wearing the arm. This can be challenging for the human as it requires more user input and the user must remember where the arm will move to, especially while engaged in another task. This report details the development of an online controller to adapt the robot's movement based on the target hand motion to deliver the grasped object. This includes avoiding collisions with the human body and other obstacles as well as online trajectory planning of an underactuated 5-degree-of-freedom robotic arm.

Additional Keywords and Phrases: Handovers, Wearable Robotics, Motion Planning

1 INTRODUCTION

Wearable robotic arms are robots that are attached to a person that can add assistive benefits. For example, providing support when assembling cars from uncomfortable angles, holding a drink while the person is working on something, or opening a door when a person's hands are full. This report outlines the development of an online controller, a controller that actively takes in sensor data to update the position of a robot in real-time, for a wearable robotic arm, the Third Arm, for use in self-handovers, handing something to oneself. An offline controller, a controller that moves a robot to predetermined positions without using sensor data, had been previously used on the Third Arm. The online controller consists of a servo controller to move the arm, sensor integration with ROS and the OptiTrack Motion Capture system for object locations, a motion planning system to control the Third Arm, and an obstacle avoidance system.

With this system, tasks like cooking can be supported with the Third Arm as one hand can be stirring a pan while another is given an ingredient out of reach normally as in Figure 1.

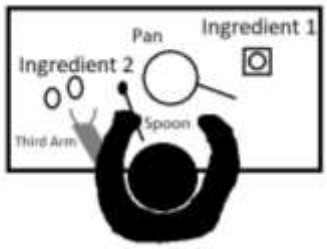

Figure 1: User wearing the Third Arm while cooking and using it to pass themselves ingredient 2, eggs, to their right hand.

2 LITERATURE REVIEW

Yang et al. [1] and Tong et al. [5] discussed the different types of wearable robotic arms and controllers developed but missing from their papers were online controllers for obstacle avoidance and handovers. In another paper, Chen et al. [2] found that they were able to use a potential field controller with their robotic arm to avoid obstacles, but only tested trajectories in constrained environments without humans or handovers. Vatsal et al. [4] discussed the inverse kinematics solution for the Third Arm in their technical report and implemented it in MATLAB; however, it was not tested or validated on the Third Arm. Vatsal [3] also developed the Third Arm and conducted experiments with it but used an offline controller.

3 DEVELOPMENT

The Third Arm software was originally developed using ROS Indigo on Ubuntu 14 using the Dynamixel Controllers ROS package. It had no online controller, integrated inverse kinematics controller, or sensor integration barring voice recognition for the experiments run by Vatsal [3].

Throughout the course of this project, the ROS and Ubuntu systems were upgraded, an updated control system was developed, OptiTrack sensing was added, an online controller was created, and we began development on obstacle avoidance systems.

3.1 Upgrading

We began by updating Ubuntu 14 to Ubuntu 20, ROS Indigo to ROS Noetic, and Python 2.7 to Python 3.1 The Dynamixel Controllers ROS package previously used to control the servos was not available for ROS Noetic, so Dynamixel Workbench was tested for use, but it did not have the velocity control and abstraction needed for the online motion planner. PyPot, an open source Dynamixel motor controller Python library, was used instead with great success and came with both register level and higher-level position and velocity control for the servos.

3.2 Control System Development

We began tests at the register level to control the Third Arm with PyPot. Once we found the arm could be controlled, we moved on to the higher-level positional and velocity controls and validated the arm also worked with these commands through a series of demos and tests. We then created a servo control class for positional and velocity control and added methods to initialize each controller in the motion planner as well as integration for the PyPot robot configuration. This controller was then tested and validated by running the controller on the Third Arm's servos.

3.3 Sensor Integration

OptiTrack integration was added to allow the Third Arm motion planner to know where the Third Arm base and gripper were as well as the target position for the gripper to go to by using OptiTrack markers as in Figure 2. We found that OptiTrack had compatibility problems with ROS Noetic, so we ported the Third Arm code over to ROS Melodic. We connected OptiTrack's streaming engine to the mocap_optitrack ROS package to get the quaternion and position vector for each object.

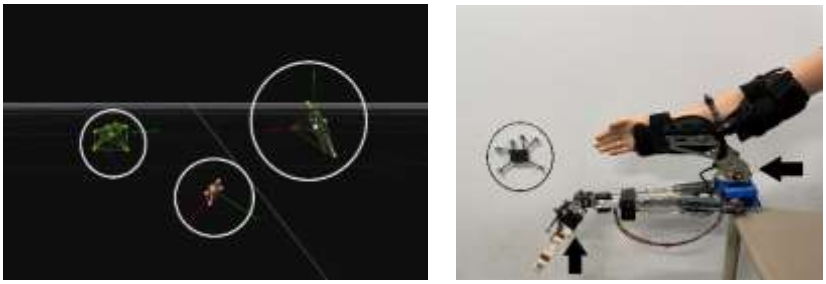

Figure 2: On the left are the OptiTrack targets set up in the software. On the right are the motion capture markers on the target and Third Arm. From left to right in both images: target, gripper, Third Arm base circled or with arrows in white and black respectively.

3.4 Motion Planning Pipeline

The motion planner allows for the Third Arm to have a feedback loop for control and use its sensor integration code to determine where, and how, to move next as shown in Figure 3. We first created a liaison file for all ROS communication that processed the quaternion and positional vectors into transformation matrices to be used by the inverse kinematics solver in the motion planner. This file would also call the motion planner and feed it the required data to actuate the arm.

We created the motion planner next. It instantiated controllers for each Third Arm servo and the PyPot robot configuration. The inverse kinematics solver was ported over to Python and used in the motion planner with the transformation data sent from the ROS liaison file. Its outputs were processed by Proportional-Integral-Derivative (PID) controllers we created for each Third Arm servo. We were able to have the arm track the target along one degree of freedom but found there were errors in Vatsal's technical report for the inverse kinematic solver and have been in the process of correcting them to have a full inverse kinematics solution.

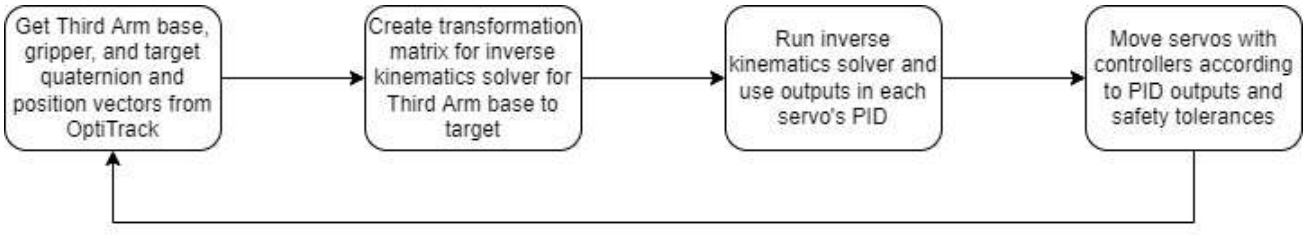

Figure 3: Motion Planning and Movement Cycle.

3.5 Obstacle Avoidance Introduction

To avoid hitting obstacles on the way to the target, obstacle avoidance will be added in the form of a potential field. The obstacle avoidance algorithm will calculate the distance vectors from each obstacle to the arm using forward kinematics to create a reference of the arm and return the location of the arm the vector corresponded to. With these vectors, the algorithm will then select the smallest one and apply an adjustment in the direction of intersection to each servo that actuates in that direction. These adjustments would then go through PID controllers before being combined with commands and safety tolerances to move towards the target.

CONCLUSIONS AND FUTURE WORK

This semester, we updated the controller for the Third Arm, added a motion planner, worked on the beginnings of an obstacle avoidance algorithm, and were able to have the arm track a target with the Third Arm base rotation.

This upcoming semester, we plan to conduct further tests to finish the obstacle avoidance system, use the servo LEDs to communicate when the arm is out of reach of the target hand, integrate voice or gesture commands to tell the arm when to perform a handover, and run an experiment with the new developments of the Third Arm.

ACKNOWLEDGEMENTS

I would like to thank Alap Kshirsagar for his collaboration and help during this project and research supervisor Dr. Guy Hoffman for his project direction and time management advice.

REFERENCES

[1] B. Yang, J. Huang, X. Chen, C. Xiong and Y. Hasegawa, "Supernumerary Robotic Limbs: A Review and Future Outlook," in *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 3, pp. 623-639, Aug. 2021, doi: 10.1109/TMRB.2021.3086016.

[2] J. Chen and K. Song, "Collision-Free Motion Planning for Human-Robot Collaborative Safety Under Cartesian Constraint," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4348-4354, doi: 10.1109/ICRA.2018.8460185.

[3] Vatsal, V. (2020). A wearable robotic forearm for human-robot collaboration (Order No. 28258774). Available from ProQuest Dissertations & Theses Global. (2484861296). Retrieved from https://www.proquest.com/dissertations-theses/wearable-robotic-forearm-human-robot/docview/2484861296/se-2?accountid=10267

[4] Vatsal, V.; Hoffman, G. Analytical Inverse Kinematics for a 5-DoF Robotic Arm with a Prismatic Joint. arXiv 2020, arXiv:2011.07286.

[5] Y. Tong and J. Liu, "Review of Research and Development of Supernumerary Robotic Limbs," in *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 5, pp. 929-952, May 2021, doi: 10.1109/JAS.2021.1003961.

A APPENDICES

Figure 3 alt-text: Four boxes with: Get Third Arm base, gripper, and target quaternion and position vectors from OptiTrack; Create transformation matrix for inverse kinematics solver for Third Arm base to target; Run inverse kinematics solver and use outputs in each servo's PID; Move servos with controllers according to PID outputs and safety tolerances.