Bumpkin: A Fist-Bumping Robot Companion

Tom Gao[1], Amy Jiang[1], Megan Lee[1], and Ranai Srivastav[1]

***Abstract*— Safe collaboration between robots (good at repetitive tasks) and humans (good at evaluating out-of-distribution events) will enable greater efficiency and better task reliability. More importantly, robots cannot always have a caged safe boundary and thus, must have the capability to understand human intention and plan accordingly. In this project, we develop a robotic arm that can identify when a human wishes to fist bump, track the closest fist, continuously tracks the target motion with the use of dynamic trajectory execution, and proceeds to bump when the fist is within a threshold distance from the tracked fist. The interaction force from the user to the robot arm is tracked, and an interaction success celebration movement is executed if the force sensed is greater than a preset threshold. The subsystems and the overall system are evaluated based on user safety, functionality, and interaction success. This is a proof of concept for more generalizable human-robot interaction, especially in cases where an articulated arm is combined with a use case involving dynamic motion tracking of human operators.**

I. INTRODUCTION

A. Background and Motivation

The safe interaction between autonomous robots and human workers has been an active area of research with numerous applications, especially when the task requires collaborative interactions between the two parties to accomplish a common goal. In common tasks such as handing over an object to a human, the most effective interaction method for the robot would be to approach the hands of the operator and signify intent to deposit an object. Separately, to mitigate safety concerns, the robot should remain compliant when perturbed by outside forces, as opposed to simply executing a preplanned trajectory.

The act of fist-bumping encapsulates both problems - tracking the hand motion of a human operator, and demonstrating compliance to outside forces - in a single collaborative goal. This project combines multiple aspects of robot autonomy in order to perform motion tracking and varying impedance control for responsive and safe fist-bumping. The project was inspired by the fluid, social nature of the fist-bumping process of the cartoon character Baymax in Big Hero 6.

Our approach makes use of the Franka Emika 6 DoF robotic arm, integrated with an Intel Realsense RGBD camera mounted on the end effector, to monitor the scene in front of the arm. Using traditional computer vision and deep learning model approaches for hand-pose detection, the camera detects and tracks the position and depth of fists within the robot's workspace, and selects the candidate fist for interaction. The interaction force and desired pose is set using the FrankaPy interface in order to plan an approaching path towards the fist, and the motion of the target is continuously updated and followed using the dynamic trajectory execution feature of the interface. A high-level planner in the form of a Finite State Machine (FSM) oversees the overall process of tracking a fist, approaching the target, and maintaining the preset amount of interaction force against the human fist. As a stretch goal, when the fist-bump is executed, the system utilizes the force and torque sensing feature to detect the user interaction. If a force threshold is met, the robot retracts the fist in a zigzag pattern as a finishing movement to signify a successful interaction.

[1]Equal contribution. All authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {zimingg,amyjiang,meganlee,ranais}@andrew.cmu.edu

II. RELATED WORK

A. Fist detection

The ubiquity of XR systems (AR, VR, and MR) have spurred ample research in the fields of hand-pose estimation. Tracking fists falls under the umbrella of hand pose estimation. Indiana University [1] and Carnegie Mellon University [2] have done significant work in collecting very diverse datasets. These bags contain both real-life and synthetic data of hand poses.

Furthermore, the task of hand tracking is robustly implemented in several state-of-the-art solutions, including the Google Mediapipe Hands solution [3] which is capable of tracking 21 landmarks of the human hand on mobile devices with real-time results (up to 60 FPS).

B. Safety-centric controls

Ensuring the safety of humans in the task-space of the robot is of utmost importance. Since we are attempting to make contact with a human being, we must do so in a way that does not endanger anyone working around the robot. Literature in the space of collaborative robots aims to ensure that robots are pliant and restrict the amount of force applied [4]. Despite the popularity of many neural-network-based methods to accomplish the same [5], classical methods such as impedance controllers [6], and as also shown in the lecture slides, provide ease of implementation.

C. Deformable Robot Systems

Deformable robots being developed at Professor Chris Atkeson's Lab at CMU gave birth to the idea of Baymax, a Disney movie protagonist robot. One of Baymax's iconic moves is to fist-bump and chime "Bah-a-la-la-la" as can be seen in https://www.youtube.com/watch?v=nC49uRsKmR8. At large, deformable robots can safely interact with the environment and respect soft and fragile items.

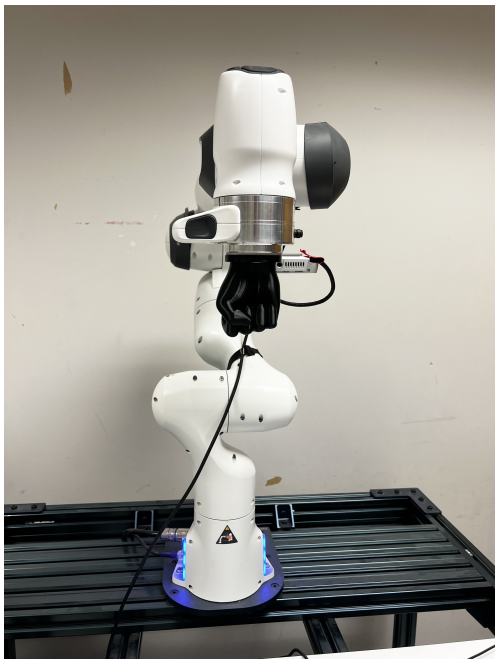

Fig. 1. Hardware Setup of the Bumpkin system.

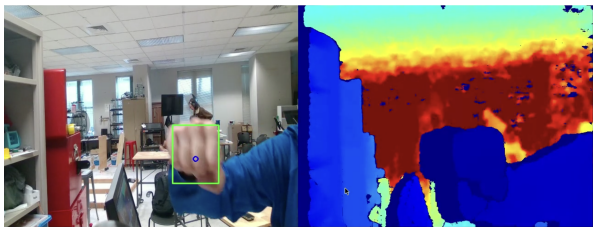

Fig. 2. A display of the camera RGB and Depth frames, overlaid with the detections from the perception subsystem.

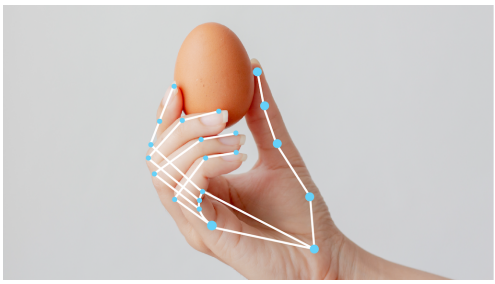

Fig. 3. An example of the output of the Google Mediapipe Hand Landmarker model.

III. METHODOLOGY

The system is composed of three primary modules: perception, motion planning, and control. The perception module identifies the closest fist in the field of the view of the camera and translates its position in the camera frame to world coordinates. The motion planning module continuously translates this desired end effector position to joint angles, plans a path from the current to the targeted configuration, and executes the plan. The control module monitors contact forces to maintain human safety when fist-bumping the user. The hardware setup and each module's implementation is discussed in detail in the following sections.

A. Hardware Setup

Bumpkin's hardware platform consists of a 6-DoF Franka arm equipped with an Intel Realsense depth sensor connected to the end effector. The end effector had been replaced with a 3D-printed fist 1. The fist end effector uses TPU material and was printed with a 25% infill to reduce the rigidity of the end-effector when it makes contact with a user's fist. This results in a more user-friendly end effector that is slightly deformable and therefore reduces risks of hurting the user during interactions. The end effector compensation has been modified on the Franka arm to 140g to account for the camera and the fist.

B. Perception

Overall, the perception module takes in RGBD frames from the end effector camera, detects fists within the frame, convert its centroid pixel location to the robot frame location, and publish this as a target point to the motion planning module.

In order to detect fists within the camera frame, we use the RGB frames from the realsense camera. As a baseline perception model, the Google's Mediapipe Hand Landmarker task model was used, which is a pretrained lightweight model capable of detecting key points of hands. The model is capable of detecting multiple hands, and providing pixel locations of 21 different key points (for every joint in the hand, from wrist to every fingertip and all joints in between) 3.

With these landmarks, the depth of the fist from the camera frame needs to be established. To do this, the pixel values of each landmark are averaged together to compute the hand centroid, which lies at the center of the fist when the fist is facing the camera.

The depth values 10x10 pixels around the hand centroid are taken, and a median depth is calculated to ensure stability of the depth estimate. These values are also verified to check the presence of zero values, which indicate invalid depths perceived by the camera resulting from objects within its 0.3m minimum range. The fist position estimate is discarded as invalid data if zero-depth pixels are detected.

With the centroid value calculated in image-pixel space, together with the median depth, we project the point onto 3D world coordinates. The Realsense camera comes with precalibrated intrinsics. This allows us to use the following fundamental projective equations:

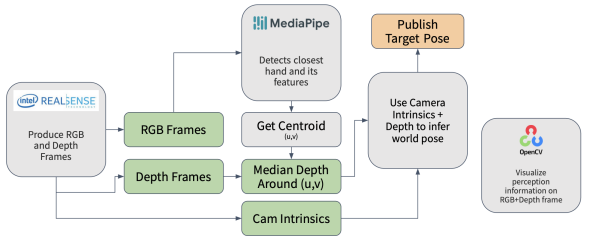

Fig. 4. An overview of the perception subsystem components and data flow.

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix} \text{ are image-pixel space coordinates}$$

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ are the world coordinates of the point in the image}$$

$$\tilde{\mathbf{p}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tilde{\mathbf{X}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \text{ are homogenized pixel/world coordinates}$$

We then have $\lambda \tilde{\mathbf{p}} = KP\tilde{\mathbf{X}}$.

$K \in \mathbb{R}^{3\times3}$ is the camera intrinsics matrix, mapping camera coordinates to pixel coordinates. These values are factory calibrated on the Realsense camera. $P \in \mathbb{R}^{3\times4}$ is the camera projection matrix, mapping world coordinates to camera coordinates and we calibrate this using `easy_handeye` or any other similar extrinsic calibration procedure. $\lambda \in \mathbb{R}^{+}$ is a constant that we use to scale using the depth values given by the depth sensor. Thus, our equations are modified using d_p, the depth value provided by the depth camera at a certain pixel as per equation (1).

$$\lambda \tilde{\mathbf{p}} = KP\tilde{\mathbf{X}} \tag{1}$$

$$P^{-1}K^{-1}d_p\tilde{\mathbf{p}} = \tilde{\mathbf{X}} \tag{2}$$

The RealSense (eye) and the end effector (hand) hand-eye calibration procedure helped map the transform between the end effector and the RealSense frame. The transform from the robot base frame to the end effector frame is calculated using forward kinematics.

Overall, the structure of the perception subsystem is outlined in 4.

C. Motion Planning

The motion planning module is responsible for planning a trajectory to reach the target point and executing that trajectory.

The desired goal pose for motion planning is divided into the end effector's desired orientation and desired position. The desired orientation is set to a fixed constant representing the appropriate lateral orientation for fist-bumping the user, which is set to face the +x direction. The desired position is set to track the detected fist location from the perception module with a given offset. In the interest of safety, the motion planning module then checks the distance between its current pose and the desired pose. It divides the trajectory to the target into linearly interpolated intermediate points if the distance is above a safety threshold. Similarly, the time duration allotted for the movement is appropriately scaled given a safety speed threshold and the distance of the movement.

The Franka arm is initialized to execute trajectories dynamically to allow for smoother motion tracking compared to planning and moving to consecutive targets independently. Hence, the target pose or poses are then published to the `FC.DEFAULT_SENSOR_PUBLISHER_TOPIC` in Frankapy [7] such that the arm will dynamically plan and move to the desired pose, taking its current position into account. The speed of the trajectory execution can be indirectly controlled by setting a maximum distance to travel per timestep, and then interpolating an intermediate goal pose that is at most this distance away from the current pose to the actual target pose.

Because of the minimum depth limitation of the RealSense camera, the motion planning module will be unable to track the desired position up to direct physical contact with the user's fist. Hence, in the case that the fist has gone out of frame, we further use the object tracking results from the perception module to determine the appropriate subsequent action by the robot. If the centroid of the fist was close to the center of the frame in several consecutive tracked frames, the robot considers itself as having adequately tracked the fist, and it has then left the frame due to the camera limitations. The robot arm will then move forward towards the user until it either meets the contact force threshold, at which the impedance controller will then soften and stop the movement, or until the maximum forward distance corresponding to the minimum sensing distance of the camera is traversed. Otherwise, the motion planning module will determine that the user's fist has left the frame and reset the robot to the home position.

D. State Machine

The system has four distinct states - Searching for a fist, tracking the fist, bumping, and celebration 5. A state machine oversees the dataflow between the perception and motion planning subsystems, and defines the interaction behavior with the target fist.

- **Searching for Fist** mode is when the MediaPipe model is running constantly and attempting to locate a fist in the received camera streams.
- **Tracking the Fist** - After a fist has been detected, the motion tracking subsystem constantly plans to and executes trajectories towards the location of the detected fist (which is consistently being updated with outputs from the perception subsystem) while discarding invalid goal poses (such as ones that result in collisions with the virtual wall).

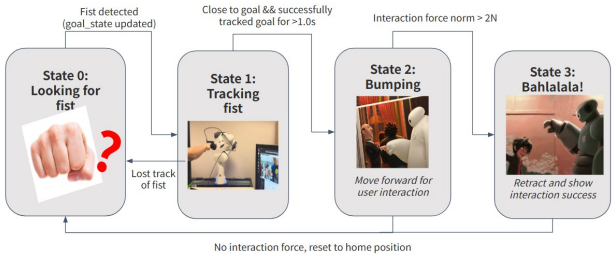

Fig. 5. An overview of the state machine and its transitions.

- **Bump**: When the arm is within 15cm of the goal fist pose, it is unable to see the fist anymore because of proximity. The robot transitions to the bump state when the current position is within 15cm of the goal position and it has successfully tracked the target position for at least 1 second consecutively. In the bump state, the end effector moves forward with impedance control, and attempts to make contact with the user fist. At this point, the dynamic trajectory execution movement is terminated, and the approach to the targeted pose is performed as a regular cartesian movement.
  Once the forward motion is completed, the robot determines if user contact is made by checking the force and torque sensors and seeing if its norm is above a set threshold. The assumption is that if the user is actively pushing against the robot with their fist, a successful interaction was made, and the state machine transitions to the next stage. Otherwise, if the force is below the set threshold, the robot retracts and returns to the first stage.
- **Interaction Success**: After the robot detects that a successful interaction was recorded, it will execute a zigzag motion back towards its home pose as an act of celebration. Once this path is executed, the robot is set back to the Searching for Fist state and begins the process again.

IV. EVALUATION

To evaluate the performance of the robot, we first tested each subsystem independently and then evaluated the system as a whole through multiple trials.

First, the perception subsystem was decoupled and tested to ensure that the subsystem was correctly aligned with the world axis. Since perception is the first subsystem in the flow, testing it decoupled did not require many changes. While the RealSense was attached to the robot and the camera to end-effector extrinsic was calibrated, the X, Y, Z location of the of the fist was obtained. We then check for the correctness of axis orientation by moving the fist in 6 directions, twice in each axis to determine positive and negative directions.

The perception subsystem was tested to ensure that only valid goal poses are published from the subsystem, and published poses must prioritize closer hands within the camera view. To this end, several tests were run and verified with the subsystem:

1) The system should not hallucinate or produce false positives of hand detections when no hands are present within the image. The subsystem was fairly robust and did not produce such detections.
2) The system should reject any hands with median depth equal to zero, which meant that the camera depth sensor is reporting invalid data. To achieve this, the aforementioned check for zero values was used.
3) The system should reject any hands detected beyond a configurable maximum distance. Since the virtual wall limited the x-axis movement of the arm to +0.5 meters, a threshold was set such that any hands more than 0.5m away from the camera center is rejected. The system successfully rejected detected hands that were in the background at the workspace and only reacted to hands that were clearly within interaction range.
4) The system should prioritize hands closer to the frame compared to further ones. Several tests were run where the operator clearly showed one of their hands but left their other hand visible within the image. While the system sometimes observes the other hand as a valid target, this is only spontaneously demonstrated when the primary hand was not detected due to lighting reasons.

The motion planning code was tested to verify desired behavior by setting up a dummy perception publisher. The publisher is designed to continuously output target points for the motion planner, where the trajectory travels between three predefined points within the workspace with a pause between each motion. The code was then tested to ensure the following:

1) The system should never exceed a movement speed that may endanger the user. As mentioned above, by setting a maximum distance and interpolating the goal point to be at maximum this distance away from the current pose, speed is indirectly controlled. The distance was configured such that there was low jerk and the end effector is easily avoidable in case of unexpected behavior.
2) The system should always converge to a stationary goal point. It was observed that the motion planner always moved to the desired point in relatively little time, and reacted to goal point changes almost immediately.
3) As we implemented our stretch goal, the system should correctly determine the force and torque being exerted by the user onto the fist. To determine a reasonable force threshold, the force sensor norm is printed while a team member interacted with the robot, which is holding a pre-determined pose. We found that a norm of 2 Newtons was enough to distinguish environmental noise and a light force of interaction from the user.

The system was then tested safely with a fake hand in the interest of safety. A picture of a fist was printed and attached to the end of a stick to maximize safety of the team members, and it is used to verify system behavior in the following tests:

Count	Status	Reason
6	Successful	Bump was successfully executed
3	Failure (Virtual Wall)	Reachable region **outside of bounds**
1	Failure (Joint Limit)	Joint 6 **moved to upper limit**

1) The system should recognize the fist picture as a valid target, and be able to track it across different camera frames. While the fist picture was finicky in some cases and caused issues with detection, the perception subsystem was able to detect the fist more than 60% of the time when it was in a valid range of the camera.
2) The system should track the motion of the printed picture and continuously adjust the arm position. Whenever the perception estimate was published, it was observed that the arm followed the movement of the picture at a safe pace.
3) The system should go into fist bumping mode when it is sufficiently close to the setpoint and when the goal pose is within the virtual wall boundaries. This was achieved whenever the prop was kept still, and the bumping was completed successfully.

After these individual tests, the system was tested by our team. After booting up the system, for each test, the team member moved their fist within detection range. In order to test the motion tracking portion of the system and to guide the robot execution to different locations, the team member moved their fist to a corner of the camera view in order to set a goal pose for the system to track. Ten trials were performed with the following results:

The system has a high success rate for reachable goals. For each of the six successful runs, the fist location in the workspace is changed, and the arms were able to converge to the goal location and move in to fist-bump the user. Note that due to the closeness threshold set for the system, some of these bumps were not completely overlapping (i.e. the robot moved more towards one side of the user's fist), but the majority of the end effector was in contact with the user.

While most of the attempts were successful, it was clear that the **definition of virtual walls were a common cause of failure of the system**. Some goal poses that are completely collision-free and which appear to be well within the reachable space of the arm triggered virtual wall collision error messages and required a relaunch of the program. It is clear that the reachable workspace of the robot arm is a limiting factor and should be taken into closer consideration when planning motion.

In one case, the sixth joint of the Franka arm moved to its maximum allowable joint value and caused the Franka control interface to shutdown. This error only occurred once during testing and deployment, and is not reproducible without a great number of runs. This may be due to the planner generating different joint space plans for each run, and in this instance, choosing a plan that crosses the maximum joint threshold value. We will not be considering this type of error as a significant source of system failure.

With our stretch goal implemented, we further confirmed that the robot was able to distinguish between a successful interaction (user applies force to fist) and a failure scenario (no force feedback), and only execute the celebration movement if the user successfully interacted with the robot.

V. CHALLENGES

There were a few challenges that came with the fist-bumping robot.

- **Virtual walls were too constricted** The biggest challenge we had was that the when presenting a fist when bumping, humans keep it very close to their body. This means that the robot has to move forward significantly in the x direction in order to achieve the goal pose and in this process, it ends up hitting the virtual walls. Sometimes, when the first position is well within the arm limits in the x-direction, the arm reaches its joint limits or hits one of the other virtual walls.
- **End effector Force Sensor Failure** Initially, the end effector does not return any force sensor value when dynamic trajectory execution was enabled. The Frankapy function in the Frankapy API returned 0 even when the arm is in contact with an object and exerting a force. To circumvent this, we relied on a distance-to-goal error at first. With further debugging, we found that this feature was only available when the arm is not actively moving, which informed our decision to stop the dynamic execution prior to sensing the force. This makes sense, since reportedly the force/torque sensor is dependent on the values of motor currents or other metrics that are only calibrated for a stationary robot.
- **Conflicting schedules between teams** Both teams worked very well with communicating schedules and how/when to change end-effectors. However, since all team members had the same mix of classes, all members had similar schedules, and this created timing conflicts. This was resolved by staggering access to the PC (mostly, our team worked earlier in the day, and coordinated with the lightsaber team's members proactively).
- Control PC crash The control PC got corrupted due to an improper switch-off procedure. This posed a significant time sink since team members were trying to fix the user PC, unaware that the issue was due to the control PC being corrupted (which is inaccessible). We spent time trying to debug ethernet interfaces through dmesg and journalctl. Since the computer was corrupted, reflashing it was the only solution. The TAs were especially helpful in fixing this issue!

VI. FUTURE WORK

Although we have been successful in reaching our goals for this system, there are still many areas for improvement that could be made.

For our stretch goals, we aimed to figure out how to read the force torque sensor values properly in order to trigger the end of the bump state, and allow the arm to retract to the home position while wiggling the end-effector in the

typical Baymax manner. Both of these stretch goals were implemented successfully.

For improvements beyond the scope of this project, the detection and tracking system could be expanded to recognize a wider variety of hand gestures, such as high-fives. The camera view could also be wider by finding a better location for it or adding an additional camera/viewpoint, such as the Azure Kinect, and using sensor fusion techniques to determine the user location even if the user's fist is blocking the field of view. This would allow a larger field of view for the robot to detect the user's hand.

Finally, the end effector could be anthropomorphic and compliant with integrated tactile sensing to improve both the safety and user experience of the fist bump. This could incorporate soft robotics principles to better mimic human hand compliance. With additional actuators acting as fingers on the fist, the interaction would become more lifelike and natural.

VII. CONCLUSION AND ACKNOWLEDGEMENT

This project represents a successful implementation of a human-robot interaction system that combines perception, motion planning, and control in a cohesive framework. Our robotic system demonstrates the ability to detect human hand gestures, track them in real time, and respond with appropriate physical interaction while maintaining safety constraints.

Throughout this project, we have explored a Mediapipe model to identify fists in the camera frame to feed into the motion planning stack. Then, we took the detected fist centroids to dynamically plan a trajectory to it with some offset. Finally, we moved slowly towards the user's fist for a fist bump with impedance control for compliant movement.

In conclusion, this project demonstrated an intuitive, safe, and engaging human-robot interaction. While fist bumping remains a very specific task, it cements the fundamentals of all human-robot interaction concepts, deciphering intention, locating points of interest, planning while adhering to external constraints, and acting out the plan but in a manner that will not endanger human safety.

The team thanks the Robotics Institute at Carnegie Mellon University for providing the resources and tools necessary to implement and learn from this project. We also thank Professor Oliver Kroemer for his knowledge and guidance throughout the 16-662 Autonomy course. Lastly, we would like to express our gratitude to our teaching assistants, Shrudhi Shanthi and Madhusha Goonesekera for their mentorship and support throughout this project.

REFERENCES

[1] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[2] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, *Hand keypoint detection in single images using multiview bootstrapping*, 2017. arXiv: 1704.07809 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1704.07809.

[3] F. Zhang *et al.*, *Mediapipe hands: On-device real-time hand tracking*, 2020. arXiv: 2006.10214 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2006.10214.

[4] I. I. F. of Robotics, *Collaborative robots - how robots work alongside humans*, 2024. [Online]. Available: https://ifr.org/ifr-press-releases/news/how-robots-work-alongside-humans.

[5] T. Brito, J. Queiroz, L. Piardi, L. A. Fernandes, J. Lima, and P. Leitão, "A machine learning approach for collaborative robot smart manufacturing inspection for quality control systems," *Procedia Manufacturing*, vol. 51, pp. 11–18, 2020, 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021), ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.10.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920318588.

[6] W. Khalil and É. Dombre, *Modeling, identification and control of Robots*. Kogan Page Science, 2006.

[7] K. Zhang, M. Sharma, J. Liang, and O. Kroemer, "A modular robotic arm control stack for research: Franka-interface and frankapy," *arXiv preprint arXiv:2011.02398*, 2020.