Technological Feasibility

Date: 04/01/2025

HapTech

Project Sponsor: Dr. Reza Razavian

Project Mentor: Veerendernath Surendernath Komala

Team Members: Landon Coonrod, Matthew Gardner, Peter Hilbert, Karissa Smallwood

HAPTECH

HAPTIC RENDERING

Table Of Contents

Introduction

Haptic technology is transforming the way humans interact with machines. By bridging the physical and virtual worlds, it enables users to experience force and tactile feedback, creating the sensation of physically interacting with digital objects. Haptic technology comes in various forms, from touch-sensitive surfaces that simulate textures to wearable devices that provide pressure and vibration feedback. This project focuses on graspable haptic technology, such as joysticks and robotic arms, which offer resistance and force feedback. These devices enhance immersive interaction with virtual environments and improve human operation of robots.

Graspable haptic technology has broad applications across various fields, including bomb disposal, space exploration, and medical training. Where robots equipped with haptic technology are used to either simulate or actually perform critical tasks with precision and safety. For example, medical students can practice procedures on virtual patients with haptic technology providing realistic sensations of incisions and suturing, without risking the well-being of a real person. In addition, robots are regularly used to dispose of toxic substances and explosives, which are remotely controlled using haptic technology to feel any forces the robot is exposed to in real time [3].

Our team was motivated to work on this project by the technological advancements that are made possible with this technology, from making labor more productive and safe to revolutionizing medical care. These advancements are driven by the growing research area of human-robot interaction, which Dr. Reza Sharif Razavian, our project sponsor, is involved in. Dr. Razavian is an assistant professor of mechanical engineering at Northern Arizona University, where his research lab, Raz Lab, integrates robotics, neuroscience, and biomechanics. One of his key research areas involves using the Franka Research 3 (FR3) robot [Fig. 1] in haptic studies, where he is leveraging CHAI3D, a haptic rendering engine, to create immersive haptic experiences. For example, a user could control a ball with the robot's arm, and the robot provides realistic force feedback, simulating collisions between the ball and other objects. However, the current setup presents several challenges. The system lacks modularity, requiring researchers to manually hard-code experiments they want to run. Additionally, there is no built-in data collection mechanism to analyze user interactions, making it difficult to extract meaningful insights from experiments. Overall, the current setup makes the research workflow inefficient and does not meet their needs.

To address these challenges, we envision a comprehensive software system that streamlines human-robot interaction studies using the FR3 robot. This system will ensure seamless integration between the haptic rendering engine and the robot, allowing for smooth and responsive haptic experiences, and allow researchers to easily configure experiments and collect interaction data. The key features of this solution include:

- A user-friendly interface that allows researchers to easily set up and run experiments, minimizing the need for manual coding.
- A robust data collection and storage system, ensuring that valuable interaction data is captured and made readily available for analysis.
- Robust integration of the Franka robot and CHAI3D for responsive force feedback.
- A remote test environment for efficient development and testing.

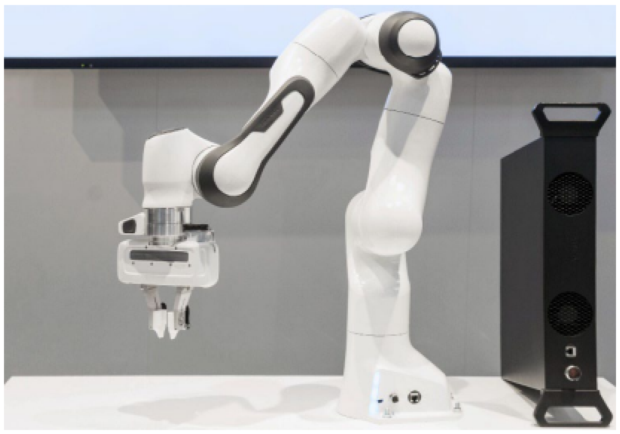

Figure 1 (Franka Research 3 Robot)

At this stage of the project, we are focusing on identifying and addressing key technical challenges required for implementing this system. Our approach involves evaluating potential solutions, comparing their feasibility, and selecting the most effective option. In this Technological Feasibility Analysis document, we begin by outlining the technical challenges we have identified, followed by an in-depth examination of each issue in the subsequent subsections. For each challenge, we explore alternative solutions, detail our investigative process, and justify our final selection.

Technological Challenges

As we begin designing and implementing our system, there are several key technical challenges that we will need to address:

1. **User-Friendly Interface:** We will need to provide a user-friendly interface for setting up and running simulations, eliminating the need for hard-coding. The challenge will be implementing an interface that provides a reasonably easy and intuitive way to add and manipulate objects to the 3D environment and configure settings.
2. **Data Collection and Storage:** We will need to implement a way to collect data from the robot during interactions and store it for later analysis. This will involve figuring out the mechanism by which we will collect the data, as well as how it will be stored.

3. **UI and CHAI3D Communication:** We will need a way for the UI and CHAI3D to communicate, allowing for environment manipulation, experiment control, and data collection. This will involve finding a suitable way to handle the networking involved and represent the data that is needed.
4. **Robot and Haptic Engine Integration:** In order to provide smooth and responsive haptic experiences, we will need to maintain and thoroughly test the integration between libfranka, the robot's API, and CHAI3D, the haptic rendering engine.
5. **Logistical Limitations:** Currently, in order to properly test CHAI3D programs, we need to be in the lab with the robot, presenting a logistical challenge. To ease the development and testing process, we need to find a way to test on our own without the robot, minimizing the need to schedule dedicated lab time.

Technology Analysis

User-Friendly Interface

Introduction

We need to provide an intuitive interface to allow fast alteration and control to the research environment.

Desired Characteristics

1. User friendliness – Efficiently displays all capabilities and allows the user to interact with the systems intuitively.
2. Modularity – Allows users to later add features to match their specific requirements. The system should be modular and easily modified.
3. Accessibility – The systems should be accessible by anyone with the Linux operating system.

Alternatives

1. Integrated Graphical Dashboard: Used to embed the CHAI3D rendering window so users can see real-time simulation and updates. This plan could include visual feedback and allow for a control panel.
2. Terminal-based Interface: GUI elements could utilize a simple terminal that can run on any Linux real time operating system that would allow users to modify specific commands. We could also do a hybrid approach that can use both the terminal and a minimal GUI for 3D rendering.
3. Modular, Multi-View Environment: A tabbed interface that can correspond to different functions

Evaluation

1. Integrated Graphical Dashboard - To evaluate the effectiveness of this approach for simulation visualization and user interaction, we researched its capabilities through product documentation and user experience reports. Based on the information, we found the following:
    - User friendliness - Intuitive and visually interactive with an embedded real-time simulation.
    - Modularity - Easily updatable components that can allow for future expansion.
    - Accessibility - Centralized control accessible to users with a wide range of technical skills.

2. Terminal-Based Interface - To evaluate the potential effectiveness of this approach and its implications for direct command-line control and streamlined operations, we examined its design specifications and gathered feedback from experienced technical users. Based on our analysis, we observed the following:
    - User friendliness - Direct command-line control with a minimal GUI for a more streamlined operation.
    - Modularity - Lightweight design that allows for quick modification.
    - Accessibility - Best suited for advanced technical users.

3. Modular, Multi-View Environment - To evaluate the potential effectiveness of this approach and its implications for the simulation setup, monitoring, and data analysis, we reviewed its modularity and interface flexibility. Based on our research, we identified the following:
    - User friendliness - Clear separation of functions using tabs to simplify navigation.
    - Modularity - Organized into independent modules that can be individually expanded or updated.
    - Accessibility - Offers flexibility in simulation setup, monitoring, and data analysis for a broad range of users.

Chosen Approach

The Integrated Graphical Dashboard offers real-time simulation that is embedded in the GUI. The GUI offers a control panel, visual feedback, and is highly accessible for users with various technical backgrounds. The terminal-based interface provides direct command line control with a minimal GUI, while lightweight, it is suited for advanced users and lacks the visual aspects. Lastly, the Modular, Multi-view Environment uses a tabbed approach to separate functions, offering high modularity but can add complexity for users unfamiliar with the navigation style.

Requirement	Integrated Graphical Dashboard	Terminal Based Interface	Modular, Multi-view Environment
User Friendliness	5/5 Its interface is intuitive and visually appealing.	3/5 Less intuitive design for non-technical users.	5/5 Easy to understand and versatile layout.
Modularity	3/5 Components are less flexible.	5/5 Offers high customization.	4/5 Decent modularity but not top-tier customization.
Accessibility	4/5 Generally easy to use with a few minor limitations.	5/5 Robust compatibility with assistive tools.	4/5 Minor navigational challenges that slightly reduce ease of use.
Total	14/15	13/15	13/15

After some consideration, we chose the terminal-based interface because its high modularity and excellent accessibility make it a flexible and inclusive environment, even though it may not be as visually intuitive as a full graphical dashboard.

Proving Feasibility

The feasibility of the terminal-based interface is proven by successful prototype tests that confirm its strong modularity and high accessibility. The command-line approach integrates seamlessly with our existing system, supports interaction, and accommodates advanced customization. This demonstrates that it can reliably meet our functional and performance requirements.

Data Collection and Storage

Introduction

There needs to be a way to seamlessly collect data from the robot during user interactions and store it for analysis purposes. The data that is collected will be used for research, therefore the storage solution must handle this efficiently.

Desired Characteristics

1. Efficiency – The system should be able to work fast and handle a large amount of data without slowing down.
2. Scalability – It should be able to handle more data as well as different types in the future.
3. Reliability – The system should try to avoid losing data and handle interruptions.
4. Accessibility – It should be easy to find and use the collected data.

Alternatives

1. Kafka: A tool used for handling large amounts of real-time data with low latency. It is a distributed event streaming platform that is good for high-throughput environments. Could possibly use the library connected to this platform, librdkafka. This library is used for simple interactions of sending and receiving, overall a lighter option.
2. PostgreSQL: An open-source database system that sends data to a server and stores it in a database. It can be used to build a REST API that CHAI3D and libfranka interact with, sending the data through HTTP requests.
3. Local File Storage (CSV): A simple, accessible way to store and share data. CSV is supported by nearly every data analysis tool. The data is saved directly to CSV files, which can then be opened in Excel.

Evaluation

1. Kafka - To evaluate the potential effectiveness of this approach and its implications for data collection and storage, we researched its capabilities by reading through multiple resources online and reviewing its features. Based on the information, we found the following in regards to our requirements:
    - Efficiency - Fast and optimized for high-throughput real-time data.
    - Scalability - Handles large-scale data streams, but that would require infrastructure setup, like adding brokers to a cluster.
    - Reliability - Fault tolerant with replication, ensures data is delivered even when faced with failures. It does require additional management.
    - Accessibility - Has authentication features like access control lists (ACLs) but requires configuration.

2. PostgreSQL - To evaluate the potential effectiveness of this approach and its implications for data collection and storage, we researched its capabilities through documentation, looked at how it handled data in different scenarios, and considered how it would integrate with Franka. We found the following:
    - Efficiency - Moderate performance, depends on API request handling.
    - Scalability - Offers both vertical and horizontal scalability, which is suitable for growing data.

    - Reliability - Protects against some data corruption but not against correctable memory errors.
    - Accessibility - Uses roles to manage database access permissions. Also SQL queries make the data retrieval easy to do.

3. Local File Storage (CSV) - To evaluate the potential effectiveness of this approach and its implications for data collection and storage, we researched its capabilities through online resources and considered if it could meet the client's needs. Based on our research, we found:
    - Efficiency - Fast data access but slow in comparison to other options.
    - Scalability - Limited scalability, it is difficult to increase the storage capacity as the needs of the data grow.
    - Reliability - Limited reliability, has strong data control, but since it saves locally there are risks.
    - Accessibility - Directly accessible from the devices it is stored on, with no need for an internet connection. However, it is limited to physical locations only.

Chosen Approach
While Kafka provides high performance and scalability, it requires infrastructure setup, which is complex and unnecessary for this project. PostgreSQL offers moderate performance and good accessibility, but it's not what our client is looking for. Local File Storage (CSV) is a very simple, user-friendly approach making it easy to use no matter the skill level.

Requirement	**Kafka**	**PostgreSQL**	**Local File Storage (CSV)**
Efficiency	5/5 High throughput, low latency, very fast.	4/5 Moderate, depends on the load of the API.	3/5 Fast data access but is limited compared to other options.
Scalability	4/5 Handles large-scale data.	4/5 Offers both vertical and horizontal scaling.	3/5 Limited by file system constraints.
Reliability	4/5 Built-in fault tolerance.	3/5 Protects against some data corruption.	3/5 Strong data control, but may be prone to risks.

Accessibility	3/5 Has good authentication features, but requires configuration.	4/5 Uses roles and has SQL to ensure easy querying.	4/5 Widely supported and directly accessible through physical locations.
Total	16/20	15/20	13/20

After some consideration, we decided that Local File Storage (CSV) is our chosen approach for data collection and storage. Despite its limitations, the client specifically requested it for this solution. CSV files are easy to use, simple, flexible, and quick to implement. Although CSV may lack some of the desired requirements, if problems keep arising, transitioning to a more suitable solution remains an option.

Proving Feasibility
Create a prototype that can collect data and write it to CSV. Then, ensure that the files can handle large-scale data without failing. Share the findings with the client to gather feedback.

UI and CHAI3D Communication

Introduction
The UI is intended to be operated on a separate device from the one running CHAI3D. This means that there needs to be a way for the UI to send commands over the network to CHAI3D to manipulate the environment and control the experiment, as well as to receive the interaction data from the experiment.

Desired Characteristics
1. Efficiency – Communication should use minimal bandwidth and have low response times.
2. Modularity – It should be easy to add new commands/features without major code changes.
3. Complexity – The chosen technology should be easy to develop and maintain.
4. Security – Remote command execution must be protected against unauthorized access.

Alternatives
1. There is an existing setup for the Delta robot, a similar robot to Franka, which uses a publish/subscribe design to receive and handle commands and could send haptic data back. Messages are manually parsed and handled via TCP sockets [4].

2. gRPC - A Remote Procedure Call (RPC) framework, which uses Protocol Buffers (Protobuf) for structured message definitions. It also supports streaming, which could be used for real-time data collection.
3. ZeroMQ - A networking library that supports many different patterns, like fan-out, publish/subscribe, and request/reply. This could be used both for commands and data collection.

Evaluation

1. Existing Delta robot setup - To evaluate the potential effectiveness of this approach, we analyzed code that our client recommended for us to look at. We took time to understand how it works and what the implications would be for our system, should we choose to go this route. From this analysis, we found the following in regards to our requirements:
    - Efficiency - Fast and efficient, needed data is sent directly via TCP sockets, little to no overhead.
    - Modularity - Not very modular; each command requires its own struct and switch case, a lot of repeated code.
    - Complexity - This approach requires manual parsing of messages. The code is very spread out, hard to tell how it works without in-depth analysis.
    - Security - Not really any security measures in place, could be prone to exploits.

2. gRPC - To evaluate the potential effectiveness of this approach and its implications for our system, we researched the capabilities of this framework by reading online resources, reading official documentation, and successfully setting up a simple trial. From these activities, we found the following in regards to our requirements:
    - Efficiency - Used for its high-performance capabilities. Request/reply: Protocol Buffers, HTTP/2: slight overhead but still fast. As for data collection, fast streaming.
    - Modularity - Messages and RPC methods are defined in a .proto file, automatically generating clean and language-agnostic code.
    - Complexity - Slight learning curve in using the framework, somewhat difficult to install and compile, but it eliminates the need for manual parsing of messages.
    - Security - gRPC offers built-in security features, such as authentication, which could be used to implement things like user roles if desired.

3. ZeroMQ - To evaluate the potential effectiveness of this approach and its implications for our system, we researched the capabilities of this library by reading online resources, reading official documentation, and successfully setting up a simple trial. From these activities, we found the following in regards to our requirements:
    - Efficiency - Works directly over TCP, very fast, no overhead from protocol buffers.

- Modularity - Does not natively enforce structured APIs, but modularity could be designed in.
- Complexity - Was very easy to set up compared to gRPC, but it requires manual message handling.
- Security - No built-in security features.

Chosen Approach

In summary, while the Delta robot setup provides a solid starting point and is efficient, it is quite complex and potentially insecure. On the other hand, gRPC is a tried and tested framework with high-performance capabilities and security features, despite the overhead involved with Protocol Buffers. It would allow for strong modularity, but there could be a slight learning curve to it. Finally, ZeroMQ is an extremely fast and lightweight networking library, but it would require extra design to ensure modularity and does not have any built-in security features. The table below shows a structured ranking of each alternative.

Requirement	**Delta Robot Setup**	**gRPC**	**ZeroMQ**
Efficiency	5/5 Very fast (direct TCP).	4/5 High-performance, but Protobuf adds slight overhead.	5/5 Very fast (direct TCP).
Modularity	2/5 Hard-coded message handling.	5/5 Auto-generates structured API, which can be used in most modern languages.	3/5 No built-in structured API, but it could be designed.
Complexity	2/5 Hard to maintain (manual parsing, scattered code).	3/5 Some learning curve, but there is extensive documentation and support.	4/5 Easier to install but requires manual message handling.
Security	1/5 No built-in security features.	5/5 Built-in TLS & authentication features.	1/5 No built-in security features.
Total	10/20	17/20	13/20

From this analysis, we have decided that the gRPC framework best suits our needs for the networking requirements of our system. It provides a modular, structured API while still providing efficiency and performance. It also provides built-in security features, which could prove to be useful. Although there is a learning curve involved, it is a popular, well-documented framework, so there is necessary support.

Proving Feasibility

To prove the feasibility of this approach we will develop a simple prototype to ensure that the communication required for experiment setup and data collection performs well enough for our client's needs, and we can fall back to a simpler TCP approach if necessary.

Robot and Haptic Engine Integration

Introduction

To create an effective haptic experience, libfranka, the robot's API, must be seamlessly integrated with CHAI3D. This integration ensures that force feedback is accurately applied in response to user interactions in the virtual environment through the robot. However, this requires real-time responsiveness and stability from both applications.

Desired characteristics

1. Real-time responsiveness – The system must process force feedback without noticeable delay.
2. Reliability – The integration should be stable across multiple experiments.
3. Scalability – The solution should allow for future extensions or modifications.

Alternatives

Since libfranka and CHAI3D must be used, this section is not on finding alternatives to these libraries, but thinking of different approaches for integration.

1. Direct API Calls - Establish direct function calls between libfranka and CHAI3D, passing data in real time.
2. Middleware - Develop an intermediate layer to translate and synchronize data between the two APIs.
3. Asynchronous Processing - Implement an event-driven system where force feedback commands are queued and processed asynchronously.

Evaluation

1. Direct API Calls - To evaluate this approach to the interaction, we have analyzed existing code our client has already attempted to develop and researched into the documentation on the API calls. We have also done a simple calculation discussing the runtime of calling directly.

- Real-time responsiveness - High responsiveness, but requires careful synchronization.
- Reliability - Prone to errors if APIs are not aligned, but reliable if done correctly.
- Scalability - Moderately difficult to scale, just requires more hard coding in calls, etc.

2. Middleware - To evaluate a middleware approach, we again did some basic calculations to estimate how much overhead we would be adding to the responsiveness of the robot and found that it was too high of a tradeoff for what our client desired.
    - Real-time responsiveness - Moderate responsiveness, adds latency.
    - Reliability - More stable, but adds complexity.
    - Scalability - Much easier to extend.

3. Asynchronous Processing - To evaluate this approach we again discussed and estimated the runtime of again adding a bit of overhead to the process. This overhead was estimated to be similar to the middleware, so as with the middleware, the benefits do not outweigh the loss in speed.
    - Real-time responsiveness - Moderate response rate as it must wait in queues.
    - Reliability - Stable but requires queue management.
    - Scalability - Moderate scalability.

Chosen Approach
Direct API offers the fastest and most direct communication between libfranka and CHAI3D. Requires careful synchronization between both systems, as a mismatch can lead to wrong data. The Middleware Layer improves modularity of the project, but introduces latency when the main goal of the project is to have it be as fast as possible. Lastly, for Asynchronous Processing, it has an event driven system that helps prevent delays by stopping responses that may block each other. It requires complex, careful coordination to avoid data loss or delays.

Requirement	**Direct Calls**	**Middleware**	**Asynchronous**
Real-time responsiveness	5/5 Fastest method.	2/5 Adds a lot of overhead.	3/5 Adds moderate overhead.
Reliability	4/5 Reliable, but complex.	4/5 Same reliability just adds more overhead and complexity.	4/5 Same as other methods, adds complexity but makes calls a little more reliable.

Scalability	3/5 Hard to scale, as you are manually calling everything.	4/5 Easier to scale as you only need to edit the middleware software.	3/5 Same as the direct calls .
Total	12/15	10/15	10/15

Since the project requires fast and efficient communication between the machine and computer, Direct API calls will likely be the primary approach, as they offer the fastest response time. However, this method requires careful implementation to ensure stability and synchronization. While it lacks modularity and scalability, its speed and low latency make it the most suitable option for real-time interaction.

Proving Feasibility

Prototype a simple force feedback test using the chosen methods, validate and record data from the experiment, and compare to the needs of the project

Logistical Testing Limitations

Introduction

Currently, we need to use the robot in the lab in order to run CHAI3D, presenting a logistical challenge for testing.

Desired Characteristics

1. Independence – Our solution should allow testing without being physically present with the robot.
2. Realism – The solution must accurately emulate the robot's behavior and haptic feedback.
3. Ease of integration – Our method should be straightforward to implement within the CHAI3D framework and be accessible to all of our team members.

Alternatives

1. Virtual test device - We can develop a simulated test device within CHAI3D that the user can control with a mouse or keyboard emulation of the robot's inputs.
2. Franka simulation - Research an existing simulator for the Franka robot that emulates response and integrates with CHAI3D.

Evaluation

1. Virtual Test Device – To evaluate the potential effectiveness of this approach for testing without physical hardware, we reviewed documentation and looked at internal testing scenarios. Based on our research, we identified the following:
   - Independence - Fully independent, allowing for testing without physical hardware.
   - Realism - May require extensive tuning in order to replicate the robot's behavior.
   - Ease of Integration - Likely straightforward to implement as a standalone module inside CHAI3D.

2. Franka Simulation – To evaluate this approach for emulating the Franka robot's behavior, we analyzed simulation capabilities and consulted relevant resources. Our findings indicate:
   - Independence - Provides an independent testing environment if a simulator is available.
   - Realism - Offers high realism by emulating the Franka robot's behavior depending on the simulators available.
   - Ease of Integration - May involve a deep integration process, impacting ease of integration.

Chosen Approach

Virtual test devices enable complete independent testing through the software simulation within CHAI3D. It seems to be straightforward to integrate, but it may require fine-tuning of the application to replicate the robot's behavior. Franka simulation offers a highly realistic emulation of Franka's robot behavior, including haptic feedback. This ensures independent testing however, it may be more complex to integrate.

Requirement	**Virtual Test Device**	**Franka Simulation**
Independence	5/5 It operates autonomously without external dependencies.	5/5 Like the test device, it functions independently, offering self-contained performance.
Realism	3/5 While functional, its simulated conditions are abstract compared to real-world dynamics.	5/5 It delivers high-fidelity simulations that closely mimic actual robotic behavior and physics.

Ease of Integration	5/5 Its modular design makes it straightforward to incorporate into existing workflows	3/5 Despite its realism, integrating it requires additional effort due to more complex interfaces.
Total	13/15	13/15

The CHAI3D approach enables complete, independent testing through software-based virtual test devices, making it straightforward to integrate into existing systems. While it offers flexibility and ease of integration, fine-tuning may be needed to accurately replicate the robot's real-world behavior. This method is ideal for development and controlled testing scenarios, balancing integration simplicity with simulation fidelity.

Proving Feasibility

Using CHAI3D, the system can operate independently and requires only minor tuning to closely mimic the robot's behavior, confirming that this method is both practical and effective for our testing needs.

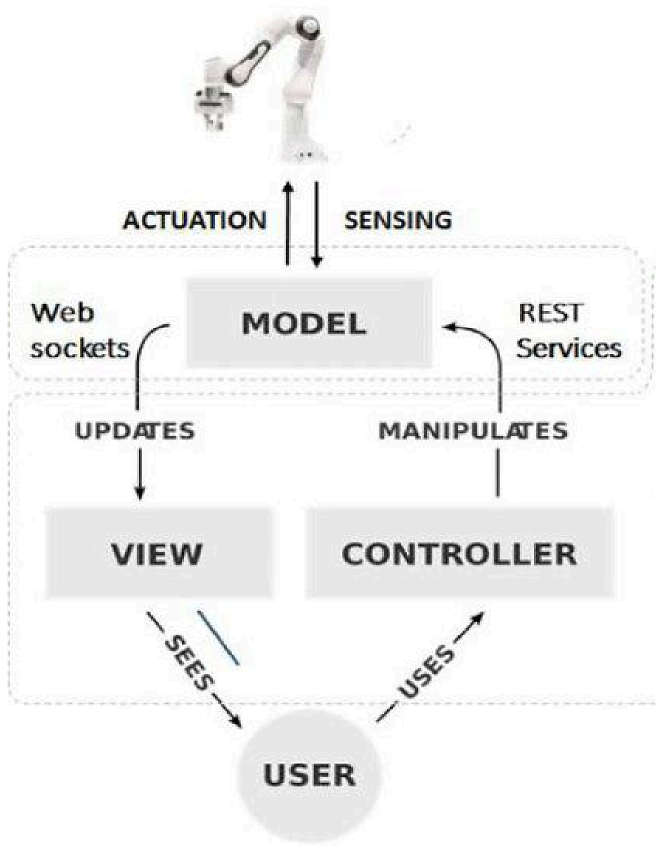

Figure 2 (Flow Chart) [2]

Technology Integration

The HapTech team faces several challenges as we work to complete this project. Fortunately, we have been provided with a set of defined languages, guidelines, and technologies that we can use to our advantage. Our task is to integrate all these elements to build a coherent and functional system that works with the Franka 3 robot. Additionally, we are responsible for collecting trial data for Dr. Reza Razavian to analyze in his lab. Together, these components define our mission.

From the program flow in Figure 2, we see that the Franka 3 robot interacts with a dynamic simulation model within a closed-loop system. This system continuously updates and maintains real-time mechanical parameters, ensuring any interaction between the robot and the model is captured. The data generated from these interactions is logged using a simple C++ script to write to a CSV file for later analysis.

The model then updates the visual display, which the user can view in the CHAI3D environment. The user interacts with the system through a basic terminal interface running on a Linux real-time operating system. This interface allows users to quickly and easily change the models that the robot interacts with. This alters the simulation and control parameters faster and more efficiently. As the user adjusts the controller, the closed-loop system updates accordingly, and the robot responds in real time.

The CHAI3D software provides visualization and haptic rendering for this project. Its advanced force-rendering algorithms, including the finger-proxy model, potential fields, and implicit-based models, enable us to develop simulations with force-feedback [1]. By using CHAI3D's C++ framework, we will expand the basic software to support multiple models that can be displayed, modified, and saved for future use. This integration ensures that the CHAI3D environment interacts directly with the robot's closed-loop system, updating mechanical parameters in real time, as shown in Figure 2.

Conclusion

In this technological feasibility analysis, we have outlined a strategy and vision for developing a software platform designed to enhance research in human-robot interaction using the Franka 3 robot. The system addresses inefficiencies by providing a modular user interface, robust data collection capabilities, seamless communication between software components, and highly responsive haptic interactions.

Throughout the analysis, we identified and explored five significant technical challenges that need to be addressed to achieve the project goals:

Major Challenges and Chosen Solutions
Throughout our analysis, we identified five key technical challenges and determined optimal solutions for each:

1. User-Friendly Interface: Terminal-based interface for a flexible and inclusive environment.
2. Data Collection & Storage: Local file storage for easy access and basic scripting.
3. UI & CHAI3D Communication: gRPC API for secure and efficient interaction.
4. Robot & Haptic Engine Integration: Direct API Calls for real-time force feedback.
5. Logistical Testing Limitation: CHAI3D Simulator through virtual test devices for remote debugging and testing.

Each solution was thoroughly evaluated through prototyping, preliminary tests, and research, confirming their feasibility and suitability for the project's requirements.

Feasibility Analysis Process
Through this analysis, we have:

1. Gained an understanding of system constraints.
2. Identified key technologies that align with project goals.
3. Designed an approach that balances performance, security, and usability.

Next Steps in Project Development
Moving forward, our immediate next steps include:

1. Developing and testing a terminal-based interface prototype.
2. Setting up and verifying the data logging system.
3. Prototyping and validating communication protocols with CHAI3D.
4. Integrating libfranka with CHAI3D to refine and test force-feedback responsiveness.
5. Exploring and evaluating the CHAI3D Simulator for remote testing.

These steps provide us with a clear and solid foundation, enabling a much easier transition into the project implementation stage as we already have a straightforward plan and alternatives in mind. By addressing these next phases, we are well-prepared to handle any emerging challenges and are confident in our ability to deliver a robust, user-centric system that significantly advances our client’s research capabilities.

References

[1] Force Dimension, "CHAI3D Software Overview," [Online]. Available: https://www.forcedimension.com/software/CHAI3D. [Accessed: 26-Mar-2025].

[2] ResearchGate, "Architecture and Screenshot of the Franka Emika Panda HPI," [Online]. Available: https://www.researchgate.net/figure/Architecture-and-screenshot-of-the-Franka-Emika-Panda-HPI_fig1_349225309. [Accessed: 26-Mar-2025].

[3] Spiceworks, "What Are Haptics?," [Online]. Available: https://www.spiceworks.com/tech/tech-general/articles/what-are-haptics/. [Accessed: 26-Mar-2025].

[4] GitHub, "Delta Robot Haptic Environment Codebase," [Online]. Available: https://github.com/rsharifr/DeltaRobot_hapticEnvironment/tree/master. [Accessed: 26-Mar-2025].

OpenAI, This document was created with the assistance of ChatGPT.