Haptic Lathe Simulator

MAE 6800: Design and Control of Haptic Systems
Fall 2024

Caleb Farrelly, Daria Kot, and Richard Zheng
Cornell University

December 13, 2025

Abstract

We present the design and implementation of a haptic lathe training device that simulates the tactile experience of operating a manual lathe. The system integrates a directly-driven handwheel with a virtual lathe environment, providing force feedback based on tool-material interaction, spindle speed, and cutting depth. Using a geared DC motor with quadrature encoder as the haptic interface and a Processing-based graphical user interface, the device enables trainees to develop muscle memory for common machining operations in a low-risk setting. The haptic rendering algorithm employs a damping-based virtual wall model derived from the Hapkit framework, with material-specific force profiles for Delrin, Aluminum 6061, Stainless Steel 316, and Inconel. This paper details the system architecture, hardware implementation, control algorithms, and force rendering equations.

Contents

1 Introduction

1.1 Motivation

Traditional lathe training requires physical machines, expert supervision, and carries inherent risks including injury and material waste. Novice machinists must develop proper "feel" for cutting operations—understanding when cuts are too aggressive, when tools are dull, and how different materials behave under the cutting tool. This tactile intuition typically requires hundreds of hours of hands-on experience.

Haptic technology offers a compelling solution: virtual training environments that provide realistic force feedback without the risks and costs of physical machining. By coupling a physical handwheel interface with a virtual lathe simulation, trainees can practice operations such as facing, turning to diameter, and boring while experiencing representative cutting forces.

1.2 Educational Objectives

The Haptic Lathe Simulator is designed to achieve the following educational objectives:

1. **Muscle Memory Development**: Users develop proper hand coordination for feed control by experiencing realistic resistance during cutting operations.
2. **Material Differentiation**: Different materials (Delrin, Aluminum, Stainless Steel, Inconel) present distinct force profiles, teaching users to recognize and adapt to material properties.
3. **Safety Awareness**: The system simulates crash conditions when the tool contacts the workpiece with the spindle stopped, reinforcing safe operating procedures.
4. **Cutting Parameter Understanding**: Users experience how spindle speed, feed rate, and depth of cut affect cutting forces, developing intuition for optimal parameters.

1.3 System Overview

The haptic lathe simulator consists of three primary subsystems:

1. **Haptic Interface**: A CQR37D geared DC motor with integral quadrature encoder, directly coupled to a handwheel that serves as both position input and force feedback output.
2. **Virtual Environment**: A Processing-based graphical user interface that renders the lathe workpiece, tool position, and material removal in real-time.
3. **Control Bridge**: A Python-based communication layer that manages bidirectional data flow between the GUI and microcontroller while computing haptic forces.

2 Background

2.1 Prior Work in Haptic Machining Simulation

Several research efforts have explored haptic feedback for machining training:

He and Chen (2006) presented the Haptic Virtual Turning Operation System (HVTOS), a prototype that simulates lathe machining with haptic, visual, and audio feedback using a PHANToM®

haptic device [1]. Their system employed spring-damper force models and NURBS-based workpiece deformation for real-time geometry updates.

Haptic milling simulators have been developed using similar principles, demonstrating that force feedback significantly improves operator training outcomes compared to visual-only simulation [2].

The Hapkit platform developed at Stanford provides an open-source framework for haptic device design and control, which informs our force rendering algorithms [3].

2.2 Virtual Wall Rendering

Virtual walls are fundamental haptic primitives that create the sensation of solid surfaces. When a user penetrates a virtual wall, the haptic device generates a restoring force proportional to penetration depth:

$$F = k \cdot x \tag{1}$$

where k is the wall stiffness (N/m) and x is the penetration depth (m). Pure spring walls can feel "buzzy" at high stiffness values, motivating the addition of damping:

$$F = k \cdot x + b \cdot \dot{x} \tag{2}$$

where b is the damping coefficient (Ns/m) and $\dot{x}$ is the penetration velocity (m/s).

2.3 Cutting Force Modeling

In actual machining, the feed force experienced by the lathe's feed mechanism is given by:

$$F_f = K_c \cdot f \cdot a_p \tag{3}$$

where:

- K_c = Specific cutting force of the workpiece material (N/mm^2)
- f = Feed per revolution (mm/rev)
- a_p = Depth of cut (mm)

This relationship guides our haptic rendering strategy: cutting forces should increase with feed rate (velocity) and depth of cut (penetration).

3 System Architecture

3.1 Architecture Overview

The system employs a three-tier architecture with clear separation of concerns:

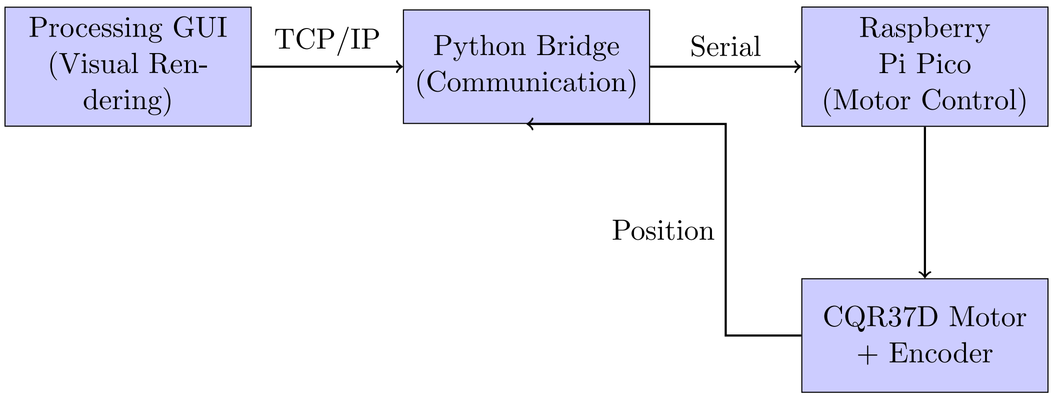
Processing GUI (Visual Rendering)
TCP/IP
Python Bridge (Communication)
Serial
Raspberry Pi Pico (Motor Control)
Position
CQR37D Motor + Encoder

Figure 1: System architecture showing data flow between components.

3.2 Communication Protocol

3.2.1 GUI to Bridge (TCP/IP, Port 5005)

The Processing GUI sends JSON-formatted commands to the Python bridge:

```
# High-speed force command (100Hz)
FORCE:<Fx>,<Fz>,<Freq>,<Yield>
# Example: FORCE:25.50,0.00,120.5,50.0

# Axis selection
{"type":"axis_select","axis":"Z"}

# Zero position
{"type":"zero_position","axis":"x"}
```

3.2.2 Bridge to Pico (Serial, 921600 baud)

The bridge sends text commands to the microcontroller:

```
spring_wall <force_N> <direction_flag> <freq_Hz> <yield_N>
status
stop
zero
```

3.3 Update Rates

The system maintains the following update rates to ensure responsive haptic feedback:

Table 1: System timing parameters.

Component	Update Rate	Purpose
Pico Control Loop	100 Hz	Motor control, force rendering
Bridge Main Loop	500 Hz	Communication management
GUI Frame Rate	60 Hz	Visual rendering
Pico Polling	100 Hz	Position feedback
Safety Checks	20 Hz	Limit monitoring

4 Hardware Implementation

4.1 Component Selection

4.1.1 Motor Selection

The haptic interface requires a motor capable of:

- Sufficient torque for realistic cutting force simulation
- Low friction for transparent backdrivability
- Integrated encoder for position sensing

We selected the **CQRobot CQR37D** geared DC motor with the following specifications:

Table 2: CQR37D motor specifications.

Parameter	**Value**
Gear Ratio	30:1
Encoder CPR (motor shaft)	64 counts/revolution
Effective Resolution	1920 counts/output revolution
Rated Voltage	12V DC
No-Load Speed	150 RPM
Stall Torque	3.1 Nm

The 30:1 gear ratio provides mechanical advantage, amplifying motor torque while reducing output speed—appropriate for the slow, deliberate movements of lathe handwheel operation.

4.1.2 Microcontroller

The **Raspberry Pi Pico** provides:

- Dual-core ARM Cortex-M0+ at 133 MHz
- Hardware PWM for motor control
- GPIO interrupts for encoder counting
- MicroPython support for rapid development

4.1.3 Motor Driver

The motor driver (H-bridge) supports:

- PWM speed control via ENA pin
- Bidirectional control via IN1/IN2 pins
- Symmetric coast mode (IN1=IN2=HIGH) for free spinning

4.2 Wiring Configuration

Table 3: Hardware pin connections.

Component	Pico Pin	GPIO	Wire Color
Encoder A	Pin 9	GP6	White
Encoder B	Pin 10	GP7	Yellow
Motor ENA (PWM)	Pin 1	GP0	—
Motor IN1	Pin 4	GP2	—
Motor IN2	Pin 2	GP1	—

4.3 Bill of Materials

Table 4: Complete bill of materials.

Item	Quantity	Cost
Raspberry Pi Pico	1	\$4.00
CQR37D Geared Motor (30:1)	1	\$34.00
Shaft Coupler	1	\$4.50
Motor Driver (H-Bridge)	1	\$15.70
3D Printed Housing	1	—
3D Printed Handwheel	1	—
Total		**\$58.20**

4.4 Mechanical Design

[PLACEHOLDER: Insert CAD renderings of the motor mount, handwheel, and housing assembly]
The mechanical assembly includes:

- 3D-printed motor mount with integrated standoffs
- 3D-printed handwheel (90mm OD) with ergonomic grip
- Shaft coupler connecting motor output to handwheel
- Housing with board mount and self-tapping thread inserts

5 Software Implementation

5.1 Motor Control (MicroPython)

The motor controller runs on the Raspberry Pi Pico, implementing real-time position tracking and force rendering.

5.1.1 Quadrature Encoder Decoding

The encoder uses hardware interrupts to track motor position:

```python
def encoder_callback(self, pin):
    a_state = self.encoder_a.value()
    b_state = self.encoder_b.value()

    if a_state != self.last_a_state:
        if a_state == b_state:
            self.encoder_count += 1   # Forward
        else:
            self.encoder_count -= 1   # Reverse

    self.last_a_state = a_state
```

Position in degrees is computed as:

$$\theta = \frac{\text{encoder\_count}}{\text{CPR} \times \text{gear\_ratio}} \times 360^\circ \tag{4}$$

For our system: $\theta = \frac{\text{count}}{64\times30} \times 360^\circ = \frac{\text{count}}{1920} \times 360^\circ$

5.1.2 Velocity Calculation

Angular velocity is computed by differentiating encoder counts:

$$\omega = \frac{\Delta\text{count}}{\text{CPR} \times \text{gear\_ratio}} \times \frac{60}{\Delta t} \quad \text{[RPM]} \tag{5}$$

An IIR low-pass filter smooths the velocity estimate:

$$\dot{x}_{\text{filt}} = 0.9 \cdot \dot{x} + 0.1 \cdot \dot{x}_{\text{prev}} \tag{6}$$

5.2 Virtual Wall Force Rendering

The haptic feedback algorithm implements a damping-based cutting force model inspired by the Hapkit framework.

5.2.1 Penetration Detection

When the tool enters the workpiece, penetration depth is computed:

$$x_{\text{pen}} = (\theta_{\text{current}} - \theta_{\text{wall}}) \cdot \text{wall\_direction} \cdot \frac{\pi}{180} \cdot r_h \tag{7}$$

where $r_h = 0.05$ m is the effective handle radius.

5.2.2 Damping-Based Force Calculation

The cutting force is computed using velocity-proportional damping, scaled by penetration depth:

$$F = -\dot{x}_{\text{filt}} \cdot c_{\text{damping}} \cdot (1 + d_{\text{scale}}) \tag{8}$$

where:

- $\dot{x}_{\text{filt}}$ = Filtered velocity at handle (m/s)
- $c_{\text{damping}} = c_{\text{base}} \times \frac{F_{\text{wall}}}{50}$ = Material-scaled damping

- $c_{\text{base}} = 2000$ Ns/m (base damping coefficient)
- $d_{\text{scale}} = \min\left(\frac{x_{\text{pen}}}{0.005}, 2.0\right)$ = Depth scaling factor

5.2.3 Torque-to-PWM Conversion (Hapkit Algorithm)

The computed force is converted to motor torque and then to PWM duty cycle using the Hapkit non-linear mapping:

$$T_p = \frac{F \cdot r_h}{\text{gear\_ratio}} \tag{9}$$

$$\text{duty} = \sqrt{\frac{|T_p|}{0.03}} \tag{10}$$

This square-root relationship linearizes the perceived force output.

5.3 Material Properties

Different materials are simulated by varying the wall stiffness k_{wall} and yield force:

Table 5: Material haptic properties.

Material	k_{wall} **(N/m)**	**Yield Force (N)**	**Color**
Delrin	50,000	25	Cream
Aluminum 6061	100,000	50	Silver
Stainless Steel 316	150,000	75	Dark Steel
Inconel	200,000	100	Bronze

The 4:1 ratio between softest (Delrin) and hardest (Inconel) materials provides clear tactile differentiation.

5.4 Graphical User Interface

The Processing-based GUI provides:

1. **Virtual Lathe Workspace**: Real-time rendering of stock geometry, tool position, and material removal
2. **Live Readouts**: X/Z position, tool load, cutting parameters
3. **Control Panel**: Material selection, axis selection, reset/zero functions
4. **Connection Status**: Visual indicator for bridge connectivity

5.4.1 Collision Detection

The tool uses a hyperbolic profile for realistic V-tool geometry:

$$y = \sqrt{(s \cdot x)^2 + R^2} - R \tag{11}$$

where s is the slope factor and R is the tool tip radius.

Collision is detected when the tool profile intersects the stock profile array at any Z-position:

$$\text{collision} = (r_{\text{tool}}(z) < r_{\text{stock}}(z)) \tag{12}$$

5.4.2 Vibration Frequency Calculation

Cutting vibration frequency is mapped from surface feet per minute (SFM):

$$\text{SFM} = \frac{\text{RPM} \times D \times \pi}{12} \tag{13}$$

$$f_{\text{vib}} = \text{SFM} \quad [\text{Hz}] \tag{14}$$

This 1:1 mapping (10 SFM = 10 Hz) provides intuitive correspondence between spindle speed and haptic vibration.

5.4.3 Material Removal

When the spindle is running (RPM ¿ 0) and the tool is engaged:

```
if (spindleRPM > 0 && penetration > yieldBuffer) {
    stockProfile[z] = min(stockProfile[z], toolRadius + yieldBuffer);
}
```

The yield buffer (1 pixel) prevents instantaneous material removal, providing a realistic "cutting feel."

5.5 Safety Features

The system implements multiple safety layers:

1. **Emergency Stop**: Spacebar triggers immediate motor stop
2. **Symmetric Coast Mode**: Both H-bridge inputs HIGH for free spin (no regenerative braking)
3. **Position Limits**: Maximum penetration of 200° prevents pass-through
4. **Heartbeat Monitoring**: 5-second timeout triggers safety stop
5. **Crash Detection**: Tool contact at 0 RPM triggers visual crash overlay

6 Experimentation

6.1 Experimental Setup

[PLACEHOLDER: Describe the experimental setup, including:

- *Physical configuration of the device*
- *Test conditions and parameters*
- *Data collection methods*
- *Participant demographics (if user study conducted)*

]

6.2 Force Rendering Validation

[PLACEHOLDER: Present data validating the force rendering:

- *Measured vs. commanded force profiles*
- *Material differentiation measurements*
- *Frequency response of the haptic system*
- *System latency measurements*

]

6.3 User Study

[PLACEHOLDER: If a pilot study was conducted, present:

- *Study protocol and tasks*
- *Participant feedback*
- *Quantitative performance metrics*
- *Subjective ratings of realism*

]

6.4 Results

[PLACEHOLDER: Summarize key experimental findings:

- *Force rendering accuracy*
- *User perception of material differences*
- *Training effectiveness indicators*
- *System reliability observations*

]

7 Discussion

7.1 Design Decisions

Several key design decisions shaped the final system:

1. **Damping-based vs. Spring-based Force Rendering**: We chose velocity-proportional damping over pure spring walls because cutting forces in actual machining are primarily determined by feed rate (velocity) rather than static position. This produces a more realistic "cutting feel."
2. **Single-Axis Interface**: While a complete lathe has two axes (X and Z), we implemented a single handwheel with software axis selection. This reduces hardware complexity while maintaining training value through the axis toggle feature.
3. **Motor as Input and Output**: Using the motor encoder as the position input (rather than a separate sensor) enables the same device to serve as both input transducer and force actuator.

7.2 Limitations

1. **Single Degree of Freedom**: Real lathe operation involves simultaneous X and Z control. The single-axis interface requires users to mentally switch between axes.

2. **No Force Sensing**: The system renders forces based on virtual geometry only; actual cutting force measurement would require additional sensors.

3. **Motor Backdrive Friction**: The geared motor introduces friction that cannot be fully compensated, affecting transparency in free motion.

7.3 Future Work

1. **Two-Axis Implementation**: Add a second motor/handwheel for simultaneous X and Z control.

2. **Force Sensing**: Integrate torque sensors for closed-loop force control.

3. **Milling Mode**: The architecture extends naturally to milling simulation as a stretch goal.

4. **Extended User Study**: Conduct a formal study comparing training outcomes between haptic and non-haptic simulation.

5. **Audio Feedback**: Add cutting sounds synchronized with spindle speed and material for multi-modal feedback.

8 Conclusion

We have presented the design, implementation, and initial evaluation of a haptic lathe training simulator. The system successfully renders material-dependent cutting forces through a directly-driven handwheel interface, enabling users to experience realistic resistance during virtual machining operations.

Key contributions include:

1. A three-tier architecture separating visualization, communication, and real-time control

2. A damping-based force rendering algorithm that produces realistic cutting feel

3. Material differentiation through adjustable wall stiffness and yield force parameters

4. Safety features including crash detection and emergency stop functionality

The system demonstrates that effective haptic machining training can be achieved with modest hardware (total cost under $60) while providing meaningful tactile feedback for skill development.

Acknowledgements

[PLACEHOLDER: Acknowledge course instructors, lab resources, and any other support received during the project.]

References

[1] X. He and Y. Chen, "A Haptic Virtual Turning Operation System," *2006 International Conference on Mechatronics and Automation*, Luoyang, China, 2006, pp. 435-440.

[2] *[PLACEHOLDER: Add proper citation for milling simulator reference from IEEE: https://ieeexplore.ieee.org/document/8590115]*

[3] *[PLACEHOLDER: Add Hapkit/Stanford haptics lab reference]*

[4] Turnin' and Burnin' Haptic Lathe Project, Lafayette College. Available: `https://sites.lafayette.edu/turnin-and-burnin/`

[5] *[PLACEHOLDER: Add VR lathe training reference from Springer: https://link.springer.com/article/10.1007/s10055-023-00816-w]*

A System Specifications Summary

Table 6: Complete system specifications.

Parameter	**Value**
Motor & Encoder	
Motor Type	CQR37D Geared DC
Gear Ratio	30:1
Encoder Resolution	64 CPR (motor), 1920 CPR (output)
Maximum RPM	150 RPM
Control System	
Microcontroller	Raspberry Pi Pico (RP2040)
Control Loop Rate	100 Hz
Serial Baud Rate	921,600 bps
PWM Frequency	1 kHz
Haptic Rendering	
Force Range	0–100 N
Damping Coefficient (base)	2000 Ns/m
Wall Stiffness Range	50,000–200,000 N/m
Handle Radius	50 mm
GUI	
Resolution	1280×720 pixels
Frame Rate	60 Hz
Stock Dimensions	4.5" L $\times$ 1.25" D
Scale Factor	60 pixels/inch