Haptic Rendering
How do you program a one-D virtual wall?
\[
\text{test } (\vec{x}_h - \vec{x}_s) \cdot \hat{u}_n = |\vec{x}_h - \vec{x}_s| |\hat{u}_n| \cos \theta = d
\]
Standard Surface Rendering in 3D

Calculate proxy position

\[ d = (\vec{x}_h - \vec{x}_s) \cdot \hat{u}_n \]

if \( d \geq r_p \)

\[ \vec{x}_p = \vec{x}_h, \quad \vec{F} = \vec{0} \]

\[ \vec{x}_h, \vec{x}_p, \vec{x}_s, \hat{u}_n, \theta \]
\[ d = (\vec{x}_h - \vec{x}_s) \cdot \hat{u}_n \]

Calculate proxy position

\[
\begin{align*}
\vec{x}_p &= \vec{x}_h - d\hat{u}_n + r_p\hat{u}_n \\
\vec{F} &= -k_s (d - r_p)\hat{u}_n
\end{align*}
\]

Limited to about 2 N/mm
Surface Properties: Hardness

Why would you want to make a wall feel harder?
How could you make a wall feel harder?

• Buy a better haptic interface.
• Perhaps try nonlinear stiffness.
• Add damping perpendicular to the plane, but only on the way in.
• Add an event-based force transient perpendicular to the plane for a short time after contact. The magnitude of the transient should scale with the magnitude of the perpendicular velocity.
A sample custom haptic device
Knob
Motor with Gearhead and Digital Encoder

Force/Torque Sensor
\[ \tau_m = k_p (\theta_d - \theta_m) + k_d (\omega_d - \omega_m) \]
\[ \Delta \theta_m \approx 1.8^\circ \]
\[ \Delta \theta_m = 1.8^\circ \cdot \frac{51200 \text{ counts}}{360^\circ} = 256 \text{ counts} \]
PostMessage(win, WM_DESTROY, NULL, NULL);

force_bias_initialize = true;

// Configure quadrature board.
ULStat = cbC7266Config (QUAD_BOARD_NUM, MOTOR_ROT, X4_QUAD, NORMAL_MODE, BINARY_ENCODING, INDEX_DISABLED, DISABLED, CARRY_BORROW, DISABLED);

// Initialize the quadrature board
LoadValue = 800000;
ULStat = cbClLoad32 (QUAD_BOARD_NUM, COUNT1, LoadValue);
ULStat = cbClLoad32 (QUAD_BOARD_NUM, COUNT2, LoadValue);
ULStat = cbClLoad32 (QUAD_BOARD_NUM, COUNT3, LoadValue);
ULStat = cbClLoad32 (QUAD_BOARD_NUM, COUNT4, LoadValue);
ULStat = cbClLoad(QUAD_BOARD_NUM, PRESCALER1, 1);
ULStat = cbClLoad(QUAD_BOARD_NUM, PRESCALER2, 1);
ULStat = cbClLoad(QUAD_BOARD_NUM, PRESCALER3, 1);
ULStat = cbClLoad(QUAD_BOARD_NUM, PRESCALER4, 1);

// Get the high resolution counter's accuracy.
QueryPerformanceFrequency(&ticksPerSecond);
sprintf(clockResult, "There are %d ticks per second", ticksPerSecond.QuadPort);

// Seed the random-number generator with current time.
srand((unsigned)time(NULL));

// Start the graphics timer
SetTimer(win, 0, GRAPHIC_UPDATE_PERIOD, NULL);

// Start the haptic thread
g_HapticThread.Start(HAPTICS_UPDATE_PERIOD, Haptic_Function, NULL);

return 0;

case WM_MOUSEMOVE:
    SetCursor(LoadCursor(NULL, IDC_ARROW));
    return 0;

case WM_DESTROY:
    // Stop the Haptic Thread
g_HapticThread.Stop();
/** Haptic Function **/
// This is the function that updates the system's forces

void __stdcall Haptic_Function(void *pW)
{
    int i;
    static double timer = 0; // Used as a timer for several different purposes.

    //////////
    /// *** TIMING ***
    //////////
    // Cache the time of the previous haptic function call.
    lastTime = thisTime;

    // Find out what time it is now. This information facilitates accurate velocity calculation.
    QueryPerformanceCounter(&thisTime);

    // Calculate time since last call in clock cycles and then convert to seconds.
    deltaTime.QuadPart = (thisTime.QuadPart - lastTime.QuadPart);
    deltaTimeS = (float) deltaTime.LowPart / (float) ticksPerSecond.QuadPart;

    //////////
    /// *** FORCE/TORQUE MEASUREMENTS ***
    //////////
    // Get present voltage values from f/t sensor
    RawVoltage(tempRawVoltage);

    // Filter voltage
    for (i=0; i<7; i++)
    {
        filteredRawVoltage[i] = lowPass((double)1.0/(2.0*PI*50.0), deltaTimeS, (double)tempRawVoltage[i], stage[i], (double)filteredRawVoltage[i]);
    }

    // Handle initialization of force/torque sensor
    if ((force_bias_initialize) && (filter_wait > 50))
    {
        Number_of_Samples++;
    
    // (CIS)--- knob_87_01_85.cpp   63% 1918   (C++ Abbrev)
// *** MOTOR CONTROL ***

// Save last position for velocity computation.
lastPosDeg = curPosDeg;

// Read in encoder signals from the QUAD44 board
ULStat = cbcIn32 (QUAD_BOARD_NUM, MOTOR_ROT, &rot_cts);

// Convert to signed counts
rot_cts_signed = rot_cts;

// Convert signed counts to degrees
curPos = rot_cts_signed / LoadValue;
curPosDeg = curPos / CTS_PER_DEG; // Converts position to units of degrees

// Check for freak position reads - if change is too much, discard this reading, and use the last one.
if (fabs(curPosDeg - lastPosDeg) > 1) {
    curPosDeg = lastPosDeg;
}

// Compute velocity and low-pass filter.
unfiltVelDeg = (curPosDeg - lastPosDeg) / deltaTimeS;
curVelDeg = LowPass1(1/(2*PI*50), deltaTimeS, unfiltVelDeg, curVelDeg);

// F/T transducer safety checks.
if (fabs(FTValues[0])>200 I fabs(FTValues[1])>200 I fabs(FTValues[2])>500 I fabs(FTValues[3])>150)
    if (fabs(FTValues[4])>1500 I fabs(FTValues[5])>2000) {
        // If over limits, make desired position present position with no output.
        desPosDeg = curPosDeg;
        desVelDeg = curVelDeg;
        current = 0;
        voltage = 0;
    } else {
        // Calculate the proxy's position and velocity during a trial for all of the different states.
        switch (state) {
            case waitingForParameters:
                break;
            case ready:
                // Trial set will start soon. Keep proxy at zero position.
                proxyPosDeg = 0;
                proxyVelDeg = 0;
                break;
            case showingCommand:
                // Next trial will start soon. Keep proxy at its current position, sitting still.
                proxyPosDeg = proxyPosDeg;
                proxyVelDeg = 0;
                break;
        }
    }

// (005) ** knob_07_01_06.cpp  69% 10001 (C++ Abbrev)
    return;
    };

    // Output the desired values to the file.
    // Write parameters.
    fprintf(output_file, "subjectNumber = %d;\n", subjectNumber);
    fprintf(output_file, "setNumber = %d;\n", setNumber);
    fprintf(output_file, "trialNumber = %d;\n", trialNumber);
    fprintf(output_file, "lineFeedback = %d;\n", lineFeedback);
    fprintf(output_file, "dotFeedback = %d;\n", dotFeedback);
    fprintf(output_file, "proprioceptiveFeedback = %d;\n", proprioceptiveFeedback);
    fprintf(output_file, "tactileFeedback = %d;\n", tactileFeedback);
    fprintf(output_file, "commandPosition = %d;\n", commandPosition);
    fprintf(output_file, "commandPosDeg = %d;\n", commandPosDeg);
    fprintf(output_file, "commandWidth = %d;\n", commandWidth);
    fprintf(output_file, "commandWidthDeg = %d;\n", commandWidthDeg);
    fprintf(output_file, "proxyAdmittance = %f;\n", proxyAdmittance);
    fprintf(output_file, "k = %f;\n", k);
    fprintf(output_file, "b = %f;\n", b);

    // Write the real time vector.
    fprintf(output_file, "clockTicksPerSecond = %lld;\n", ticksPerSecond);
    fprintf(output_file, "tClock = [";)
    for (i=0; i<dataIndex; i++) {
        fprintf(output_file, "%lld\n", timeArray[i]);
    }
    fprintf(output_file, "] = %lld;\n", timeArray[dataIndex]);
    fprintf(output_file, "t = tClock / clockTicksPerSecond;\n";)

    // Write time-varying data.
    fprintf(output_file, "docVoltage = [";)
    for (i=0; i<dataIndex; i++) {
        fprintf(output_file, "%.9f;\n", docVoltageArray[i]);
    }
    fprintf(output_file, "];\n\n";)

    fprintf(output_file, "fingerForce = [";)
    for (i=0; i<dataIndex; i++) {
        fprintf(output_file, "%.9f;\n", fingerForceArray[i]);
    }
    fprintf(output_file, "];\n\n";)

    fprintf(output_file, "motorPosition = [";)
    for (i=0; i<dataIndex; i++) {
        fprintf(output_file, "%.9f;\n", motorPositionArray[i]);
    }
    fprintf(output_file, "];\n\n";)

    "(005)"  knob_07_R1_R6.cpp  93% 11346  (C++ Abbrev)
Haptic Thread Cycle Index

Time Since Last Cycle (ms) vs. Haptic Thread Cycle Index
Haptic Virtual Environment
Haptic Remote Environment

\[ \vec{x}_h \quad \vec{F}_{cmd} \]
Teleoperation extends the reach of the human hand.
Teleoperation extends the reach of the human hand.

**Teleoperation**

Operator | Haptic Interface (Master) | Remote Robot (Slave) | Environment
---|---|---|---

Controller
Mechanical Teleoperation

Goertz, 1952
Mechanical Teleoperation
Modern Teleoperation

Kuchenbecker, 2006
Robot-Assisted Minimally Invasive Surgery

(Intuitive Surgical, Inc., 1998)
In this chapter we present an overview of the field of telerobotics with a focus on central aspects. Motivated by an historical perspective and some challenging applications of this research area, a classification of control architectures is given, including an introduction to the different strategies. An emphasis is taken on bilateral control and force feedback, which is a vital research field today. Finally, we suggest some literature for a closer engagement with the topic of telerobotics.

### 31.1 Overview

Telerobotics is perhaps one of the earliest aspects of robotics. Literally meaning robotics at a distance, it is generally understood to refer to robotics with a human operator in control or human-in-the-loop. Any high-level, planning, or cognitive decisions are made by the human user, while the robot is responsible for their mechanical implementation. In essence, the brain is removed or distant from the body. Herein the term telerobotics, which is derived from the Greek and means distant, is generalized to imply a barrier between the user and the environment. This barrier is overcome by remote-controlling a robot at the environment, as indicated in Fig. 31.1. Besides distance, barriers may be imposed by hazardous environments or scaling to very large or small environments. All barriers have in common that the user cannot (or will not) physically reach the environment.

While the physical separation may be very small, with the human operator and the robot sometimes occupying the same room, telerobotic systems are often at least conceptually split into two sites: the local site with the human operator and all elements necessary to support the system’s connection with the user, which could be joysticks, monitors, keyboards, or other input/output devices, and the remote site, which contains the robot and supporting sensors and control elements.

To support this functionality, telerobotics integrates many areas of robotics. At the remote site, to operate the robot and execute the human’s commands, the system may control motion and/or forces of the robot. We refer to Chaps. 6 and 7 for detailed descriptions of these areas. Also, sensors are invaluable (Chap. 4), including force sensors (Chap. 19) and others (Part C). Meanwhile, at the local site information is often displayed haptically (Chap. 30).

A recent addition to telerobotics is the use of computer networks to transmit information between the two sites. This is the focus of Chap. 32 and opens up new possibilities in architectures. For example, a single robot may be shared between multiple users or a single user may

### 31.3 Control Architectures

#### 31.3.1 Supervisory Control

#### 31.3.2 Shared Control

#### 31.3.3 Direct and Bilateral Teleoperation

### 31.4 Bilateral Control and Force Feedback

#### 31.4.1 Position/Force Control

#### 31.4.2 Passivity and Stability

#### 31.4.3 Transparency and Multichannel Feedback

#### 31.4.4 Time Delay and Scattering Theory

#### 31.4.5 Wave Variables

### 31.5 Conclusions and Further Reading


Provides a good introduction to the topic of teleoperation, including discussions of varying levels of remote robot autonomy and different control schemes for achieving force feedback.
Teleoperation

• Teleoperation has always been tightly intertwined with robotics, especially manipulators.

• Control system design is a primary concern:
  • Stability
  • Transparency
How do we want this system to behave?

How should we connect the sensors and actuators of the master and slave to make the system behave well?
Position-Forward Control

Haptic Interface

Desired Position

User

Master Handle

Master Motor

Remote Robot

Slave Motor

Slave Tip

Unilateral PD Controller

Environment
Position-Force Control

Diagram showing the interaction between a user and a remote robot, with control loops for position and force.
Position-Position Control

Diagram showing a control system with a Haptic Interface and a Remote Robot. The system includes:
- Desired Position
- Master Handle
- Master Motor
- Slave Motor
- Slave Tip
- Bilateral PD Controller
- Environment
With two impedance-type (backdrivable) devices, the most common controller is position-position, also known as position exchange.

Each device has a desired state (position and velocity), which is computed from measured states.

Separate controllers try to make each device achieve its desired state by using the motors to output forces.
Midterm Exam
MEAM 520, Introduction to Robotics
University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.
November 8, 2012

You must take this exam independently, without assistance from anyone else. You may bring
in a calculator and two 8.5" × 11" sheets of notes for reference. Aside from these two pages of notes,
you may not consult any outside references, such as the textbook or the Internet. Any suspected
violations of Penn’s Code of Academic Integrity will be reported to the Office of Student Conduct
for investigation.

This exam consists of several problems. We recommend you look at all of the problems before
starting to work. If you need clarification on any question, please ask a member of the teaching
team. When you work out each problem, please show all steps and box your answer. On problems
involving actual numbers, please keep your solution symbolic for as long as possible; this will make
your work easier to follow and easier to grade. The exam is worth a total of 100 points, and partial
credit will be awarded for the correct approach even when you do not arrive at the correct answer.

<table>
<thead>
<tr>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>20</td>
</tr>
<tr>
<td>Problem 2</td>
<td>20</td>
</tr>
<tr>
<td>Problem 3</td>
<td>15</td>
</tr>
<tr>
<td>Problem 4</td>
<td>20</td>
</tr>
<tr>
<td>Problem 5</td>
<td>25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

I agree to abide by the University of Pennsylvania Code of Academic Integrity during
this exam. I pledge that all work is my own and has been completed without the use
of unauthorized aid or materials.

Signature

Date
Midterm Overall

Mean = 76.6
Median = 78.0
St. Dev. = 13.2
Problem 1 (Short Answers)

Mean = 13.5
Median = 14.0
St. Dev. = 3.4
Problem 2 (Homogeneous Transformations)

Mean = 14.0
Median = 16.0
St. Dev. = 6.3
Problem 3 (Inverse Orientation Kinematics)

Mean = 9.9  
Median = 11.0  
St. Dev. = 4.5

\[ R = R_{z,\phi}R_{y,\theta}R_{x,\psi} = \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix} \]

\[ \mathcal{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \]
Problem 4 (DH Parameters)

Mean = 18.5
Median = 19.0
St. Dev. = 2.5
Problem 5 (Inverse Position Kinematics, Jacobian, Singularities)

Mean = 20.7
Median = 22.0
St. Dev. = 3.8
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic (Linked to Lecture Slides)</th>
<th>Reading</th>
<th>Assignments Due</th>
<th>Project Deadlines</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 Thu, 9/6</td>
<td>Course Logistics and Motivation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02 Tue, 9/11</td>
<td>Rotation Matrices</td>
<td>8.1, 2.1-2.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03 Thu, 9/13</td>
<td>Homogenous Transformations</td>
<td>2.4-2.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04 Tue, 9/18</td>
<td>Manipulator Kinematics</td>
<td>1.1-1.3, 3.1</td>
<td>HW61 (Flying Real)</td>
<td></td>
</tr>
<tr>
<td>05 Thu, 9/20</td>
<td>Denavit-Hartenberg (DH)</td>
<td>3.2</td>
<td>HW62 (SCARA)</td>
<td></td>
</tr>
<tr>
<td>06 Tue, 9/25</td>
<td>More Denavit-Hartenberg (DH)</td>
<td>3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07 Thu, 9/27</td>
<td>Inverse Kinematics (IK)</td>
<td>3.3-3.4</td>
<td>HW63 (PUMA FK + SCARA IK)</td>
<td>PUMA Light Painting: Teams</td>
</tr>
<tr>
<td>08 Tue, 10/2</td>
<td>More Inverse Kinematics (IK)</td>
<td>3.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09 Thu, 10/4</td>
<td>PUMA 260 and Project 1</td>
<td></td>
<td>HW64 (Jacobi)</td>
<td></td>
</tr>
<tr>
<td>10 Thu, 10/9</td>
<td>More Manipulator Kinematics</td>
<td>3.3</td>
<td>HW65 (Phantom)</td>
<td></td>
</tr>
<tr>
<td>11 Tue, 10/16</td>
<td>Velocity Kinematics</td>
<td>4.6</td>
<td>PUMA Light Painting: IK</td>
<td></td>
</tr>
<tr>
<td>12 Tue, 10/18</td>
<td>More Velocity Kinematics</td>
<td>4.6, 4.9, 4.11, 4.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 Thu, 10/25</td>
<td>From Simulation to Reality</td>
<td></td>
<td></td>
<td>PUMA Light Painting: Simulation</td>
</tr>
<tr>
<td>14 Thu, 10/30</td>
<td>No lecture - fall break</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 Tue, 11/1</td>
<td>Robot Trajectories</td>
<td>5.1, 5.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 Thu, 11/8</td>
<td>Midterm Exam (Solution)</td>
<td></td>
<td>HW64 (Jacobi)</td>
<td></td>
</tr>
<tr>
<td>17 Thu, 11/13</td>
<td>Haptic Interface Hardware</td>
<td>6.1, 6.2</td>
<td>PUMA Light Painting: Reality</td>
<td></td>
</tr>
<tr>
<td>18 Thu, 11/15</td>
<td>Telesoperation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19 Thu, 11/20</td>
<td>No lecture - Thanksgiving</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(note: all items are due at 3:00 p.m. unless otherwise specified)
Midterm Exam
MEAM 520, Introduction to Robotics
University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

November 8, 2012

You must take this exam independently, without assistance from anyone else. You may bring
a calculator and two 8.5\textquoteright\texttimes11\textquoteright sheets of notes for reference. Aside from these two pages of notes,
you may not consult any outside references, such as the textbook or the Internet. Any suspected
violations of Penn's Code of Academic Integrity will be reported to the Office of Student Conduct
for investigation.

This exam consists of several problems. We recommend you look at all of the problems before
starting to work. If you need clarification on any question, please ask a member of the teaching
team. When you work out each problem, please show all steps and box your answer. On problems
involving actual numbers, please keep your solution symbolic for as long as possible; this will make
your work easier to follow and easier to grade. The exam is worth a total of 100 points, and partial
credit will be awarded for the correct approach even when you do not arrive at the correct answer.

<table>
<thead>
<tr>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>20</td>
</tr>
<tr>
<td>Problem 2</td>
<td>20</td>
</tr>
<tr>
<td>Problem 3</td>
<td>15</td>
</tr>
<tr>
<td>Problem 4</td>
<td>20</td>
</tr>
<tr>
<td>Problem 5</td>
<td>25</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
</tr>
</tbody>
</table>

I agree to abide by the University of Pennsylvania Code of Academic Integrity during
this exam. I pledge that all work is my own and has been completed without the use
of unauthorized aid or materials.

Signature ______________________
Date ______________________
Look over your exam and compare with the solution.

If you think we made a mistake in grading your test, write out an explanation on a separate piece of paper.

Give your written inquiry and your test to Philip.

We will correct any grading mistakes.
Approximate grade breakdown

A+ 96  A  89  A-  83  B+  78  B  73  B-  66  C+  60  C  54  C-

Please make an appointment to talk with me if you got less than a 55/100.