MEAM 520

Robotic Systems: Teleoperation and Mobile Robots

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception L (GRASP)
MEAM Department, SEAS, University of Penns

GRASP LABORATORY

Lecture 27: December 10, 2013


UPENN 2013 MEAM 410/510
MECHA
TRONI
CS
#include "m_general.h"
int main(void){
while(TRUE){
m_red(TOGGLE);
m_wait(510);
}
}
Wu & Chen Auditorium
Thursday, December 12th
Final rounds starting at 6:00 PM
ROBOCKEY 2013

MEAM 520 Calendar

Tuesday 12/3 – More Motion Control
Thursday 12/5 – Haptic Rendering
Introduce Homework 9
Thursday 12/5 – Project 2 Part 2 (Sim Painting) due

Tuesday 12/10 – Teleoperation & Mobile Robots
Tuesday 12/10 – Homework 9 (Haptics) due, with no penalty up to 12/14

Ongoing – Record Light Paintings on PUMA

Wednesday 12/18 – Final Exam noon to 2 p.m.

Final Exam

Wednesday, December 18

Noon to 2:00 p.m.

Claudia Cohen G17

Comprehensive, covering everything
through today's lecture,
emphasis on material from homework and projects

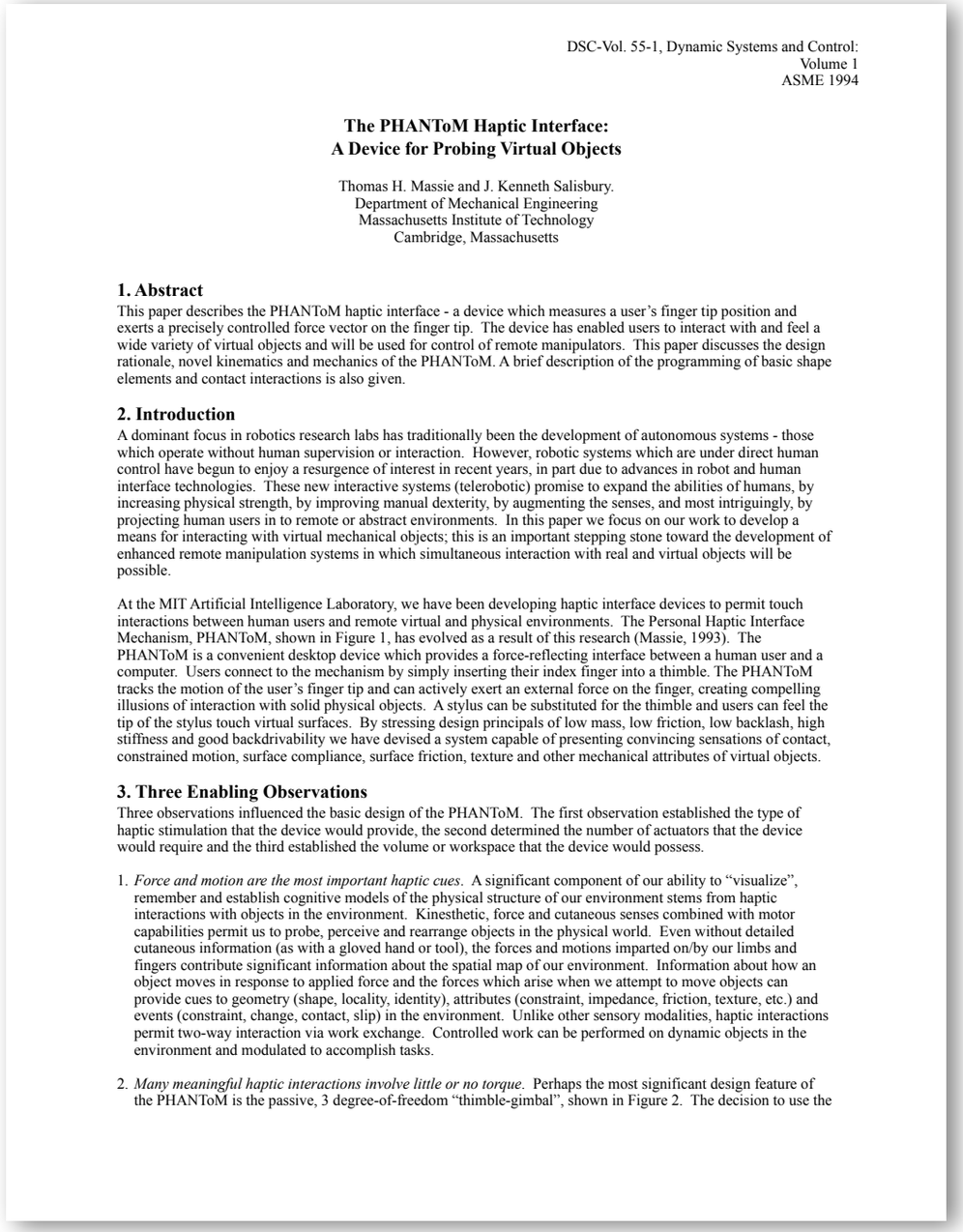
Closed book

Four sheets of notes (handwritten and/or printed)

Calculator allowed

I'll post a practice final with solutions soon.

Haptic Rendering



DSC-Vol. 55-1, Dynamic Systems and Control:
Volume 1
ASME 1994

The PHANToM Haptic Interface:
A Device for Probing Virtual Objects

Thomas H. Massie and J. Kenneth Salisbury.
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts

1. Abstract

This paper describes the PHANToM haptic interface - a device which measures a user's finger tip position and exerts a precisely controlled force vector on the finger tip. The device has enabled users to interact with and feel a wide variety of virtual objects and will be used for control of remote manipulators. This paper discusses the design rationale, novel kinematics and mechanics of the PHANToM. A brief description of the programming of basic shape elements and contact interactions is also given.

2. Introduction

A dominant focus in robotics research labs has traditionally been the development of autonomous systems - those which operate without human supervision or interaction. However, robotic systems which are under direct human control have begun to enjoy a resurgence of interest in recent years, in part due to advances in robot and human interface technologies. These new interactive systems (telerobotic) promise to expand the abilities of humans, by increasing physical strength, by improving manual dexterity, by augmenting the senses, and most intriguingly, by projecting human users in to remote or abstract environments. In this paper we focus on our work to develop a means for interacting with virtual mechanical objects; this is an important stepping stone toward the development of enhanced remote manipulation systems in which simultaneous interaction with real and virtual objects will be possible.

At the MIT Artificial Intelligence Laboratory, we have been developing haptic interface devices to permit touch interactions between human users and remote virtual and physical environments. The Personal Haptic Interface Mechanism, PHANToM, shown in Figure 1, has evolved as a result of this research (Massie, 1993). The PHANToM is a convenient desktop device which provides a force-reflecting interface between a human user and a computer. Users connect to the mechanism by simply inserting their index finger into a thimble. The PHANToM tracks the motion of the user's finger tip and can actively exert an external force on the finger, creating compelling illusions of interaction with solid physical objects. A stylus can be substituted for the thimble and users can feel the tip of the stylus touch virtual surfaces. By stressing design principals of low mass, low friction, low backlash, high stiffness and good backdrivability we have devised a system capable of presenting convincing sensations of contact, constrained motion, surface compliance, surface friction, texture and other mechanical attributes of virtual objects.

3. Three Enabling Observations

Three observations influenced the basic design of the PHANToM. The first observation established the type of haptic stimulation that the device would provide, the second determined the number of actuators that the device would require and the third established the volume or workspace that the device would possess.

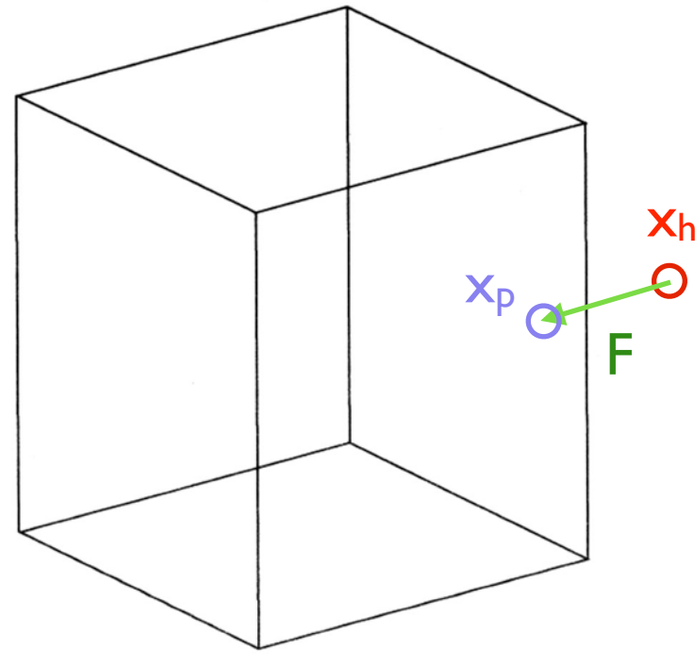
1. *Force and motion are the most important haptic cues.* A significant component of our ability to "visualize", remember and establish cognitive models of the physical structure of our environment stems from haptic interactions with objects in the environment. Kinesthetic, force and cutaneous senses combined with motor capabilities permit us to probe, perceive and rearrange objects in the physical world. Even without detailed cutaneous information (as with a gloved hand or tool), the forces and motions imparted on/by our limbs and fingers contribute significant information about the spatial map of our environment. Information about how an object moves in response to applied force and the forces which arise when we attempt to move objects can provide cues to geometry (shape, locality, identity), attributes (constraint, impedance, friction, texture, etc.) and events (constraint, change, contact, slip) in the environment. Unlike other sensory modalities, haptic interactions permit two-way interaction via work exchange. Controlled work can be performed on dynamic objects in the environment and modulated to accomplish tasks.

2. *Many meaningful haptic interactions involve little or no torque.* Perhaps the most significant design feature of the PHANToM is the passive, 3 degree-of-freedom "thimble-gimbal", shown in Figure 2. The decision to use the

PHANToM Haptic Interface

$$\vec{F}_h = f(\vec{x}_h)$$

Object contact is rendered by keeping track of a geometric proxy. It stays on the surface of the object, and we use a virtual spring to pull the user toward its location, always perpendicular to the surface.



1. Free space must feel free.

2. Solid virtual objects must feel stiff.

3. Virtual constraints must not be easily saturated.

Homework 9:
Haptic Rendering with the Phantom

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

December 5, 2013

This assignment is due on **Tuesday, December 10, by midnight (11:59:59 p.m.)** Your code should be submitted via email according to the instructions at the end of this document. To help you handle the intensity of the end of the semester, **late submissions will be accepted with no penalty until midnight on Saturday, December 14**. Submissions after Saturday will be penalized by 10% for each partial or full day late, up to 20%. After Monday, December 16, no further assignments may be submitted.

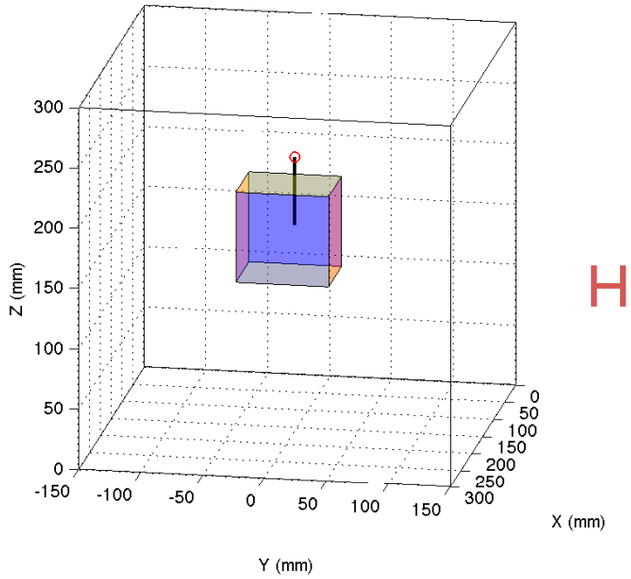
You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you write down must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Pair Programming

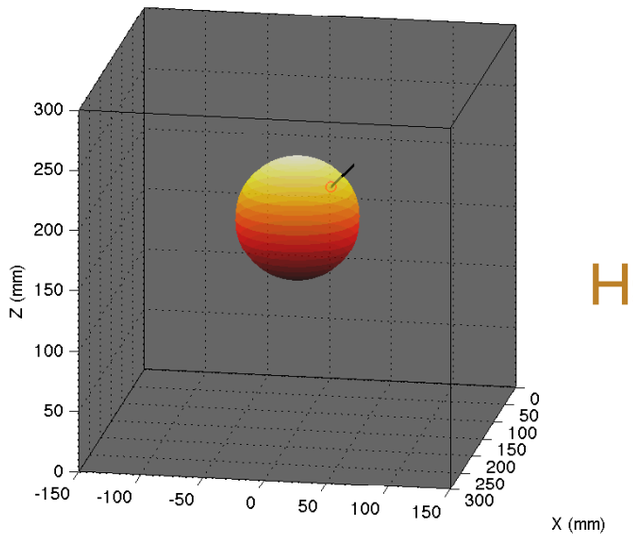
Because it requires the use of a real robot, you must do this assignment with a partner; individual submissions are not allowed. Working with a partner will help keep you safe, keep the Phantom safe, and ensure everyone has a chance to work with the robot in the time available. You may work with any other student in the class. If you're looking for a partner, consider using the "Search for Teammates!" tool on Piazza.

You should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one set of MATLAB scripts for which you are both jointly responsible, and you will both receive the same grade. Please follow these pair programming guidelines, which were adapted from "All I really need to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000:

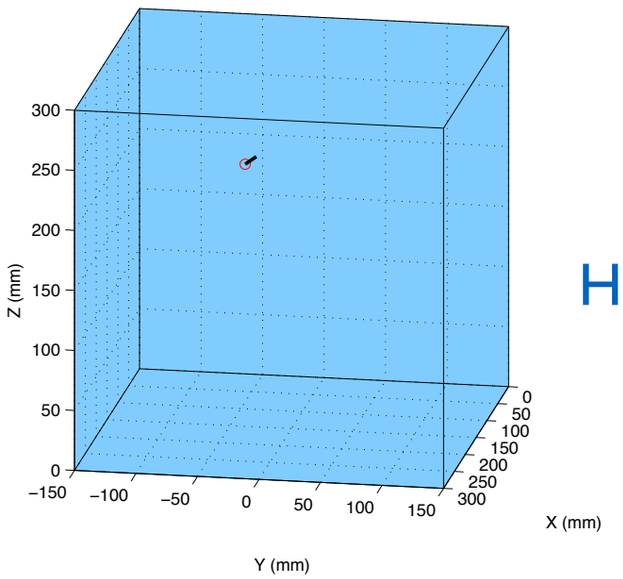
- Start with a good attitude, setting aside any skepticism and expecting to jell with your partner.
- Don't start writing code alone. Arrange a meeting with your partner as soon as you can.
- Use just one computer, and sit side by side; a desktop computer with a large monitor is better for this than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (using the mouse and keyboard or recording design ideas) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every thirty minutes, *even if one partner is much more experienced than the other*. You may want to set a timer to help you remember to switch.
- If you notice a bug in the code your partner is typing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Recognize that pair programming usually takes more effort than programming alone, but it produces better code, deeper learning, and a more positive experience for the participants.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.



Haptic Box Demo



Haptic Ball

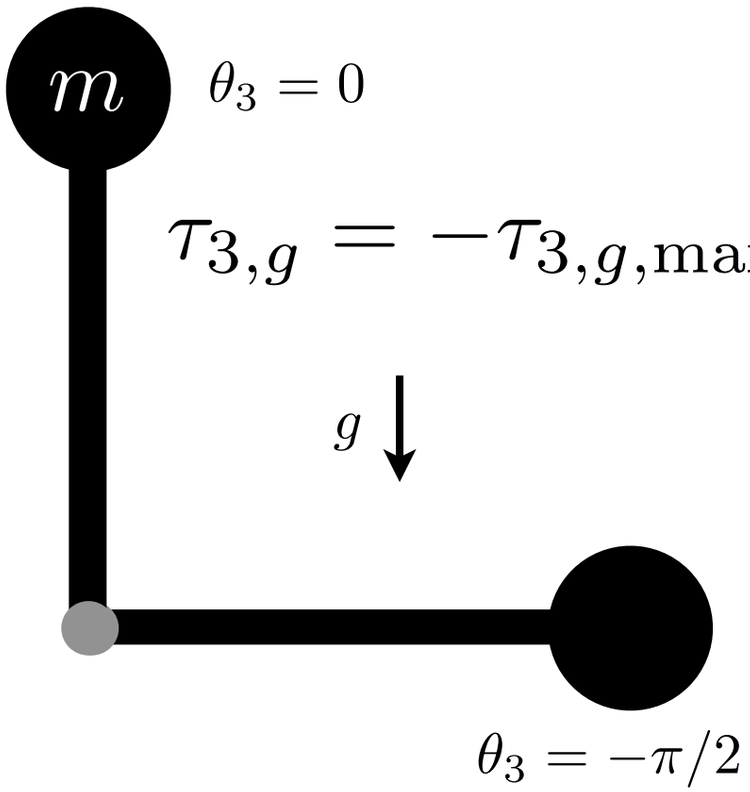


Haptic Damping

Robot Dynamics and Gravity Compensation

To improve motion tracking, engineers generally seek to minimize the inertia and friction of a robot.

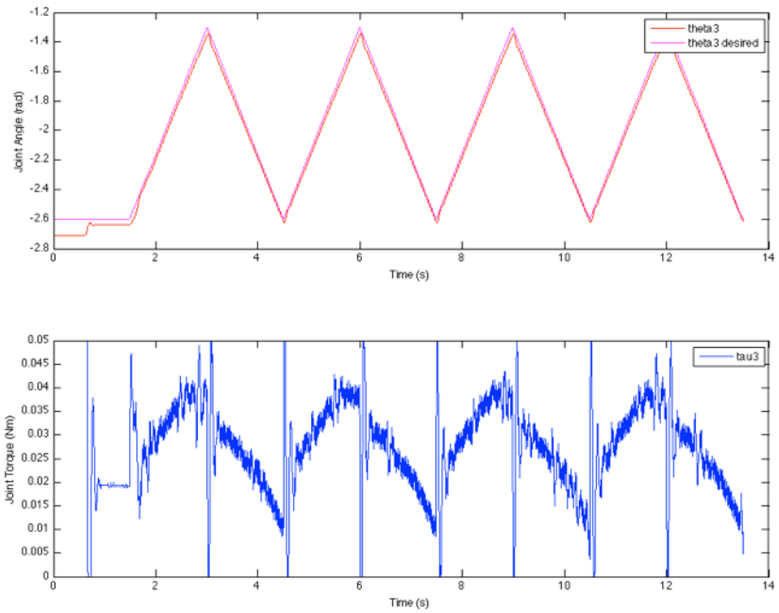
We can compensate for a robot's weight (within limits) using the robot's own motors and a simple model.



Option 1: Inspect the robot to determine the form we expect for the gravity torque, then tune the strength through experiments.

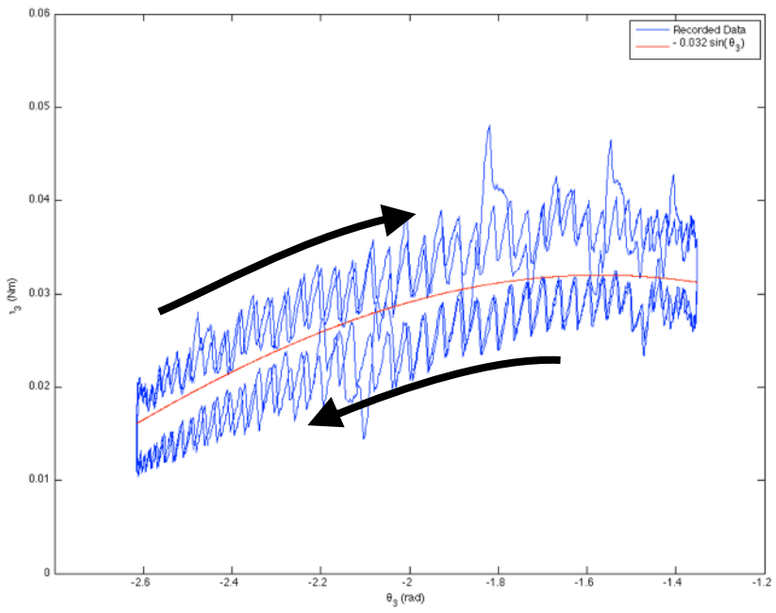
Option 2: Move the robot slowly through a trajectory and record the PD torque needed to hold up the weight of the robot.

Linear Trajectories



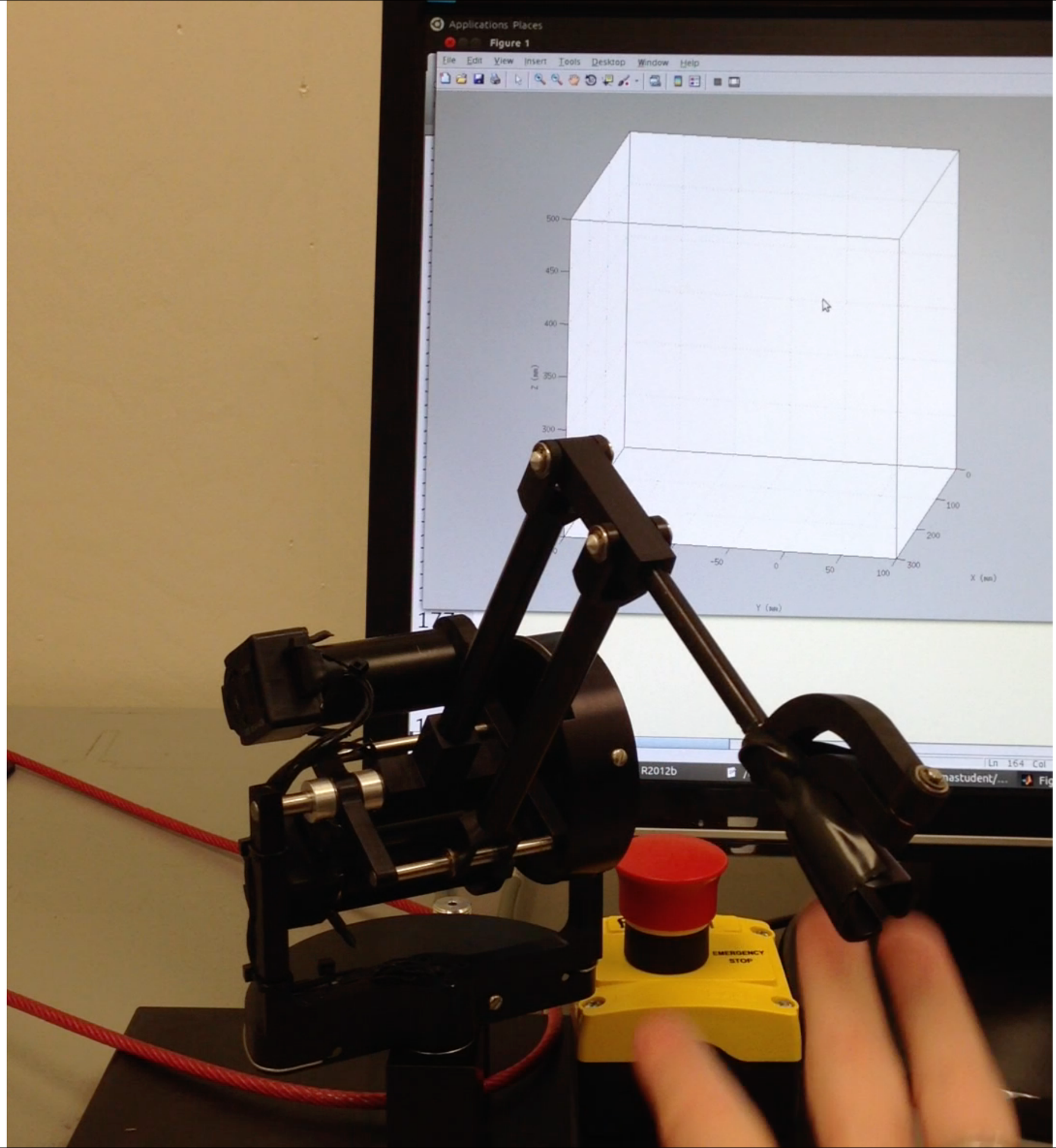
Inertia: big spikes at ends
Weight: non-zero mean at a given angle
Friction: direction-dependent offset

Cubic Polynomial Trajectories



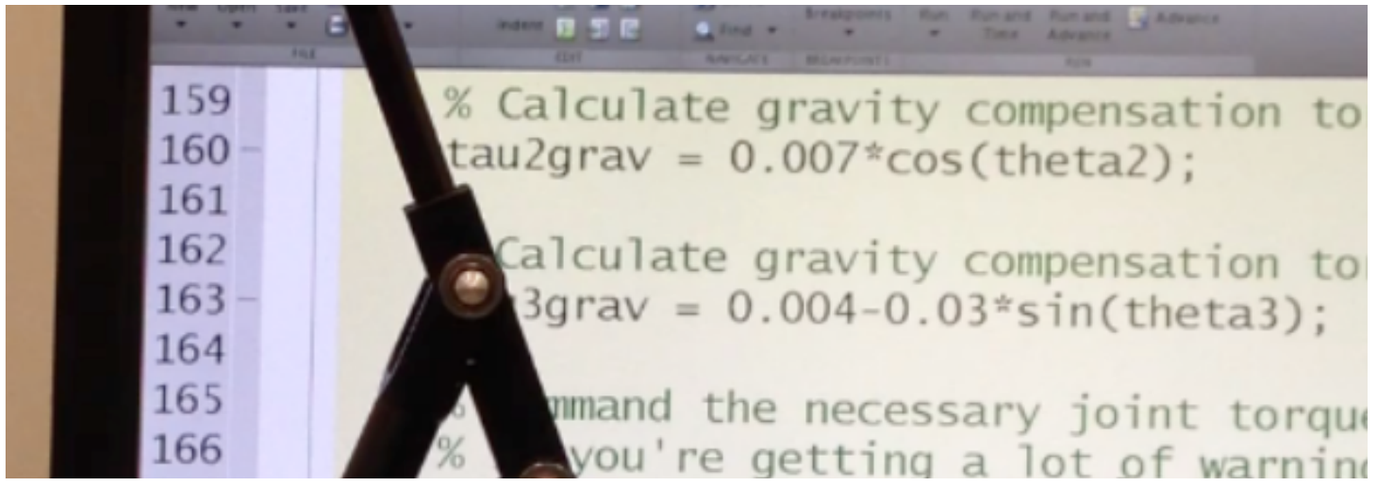
Smoother trajectory
Cleaner signal for fitting
Squiggles are timing artifact

with updated gravity compensation on joints 2 and 3



We expected gravity compensation of this form:

$$\tau_{3,g} = -\tau_{3,g,\max} \sin\theta_3$$



Experiments showed that a different function yielded the best gravity compensation:

$$\tau_{3,g} = 0.004\ \mathrm{Nm} - 0.03\ \mathrm{Nm} \sin\theta_3$$

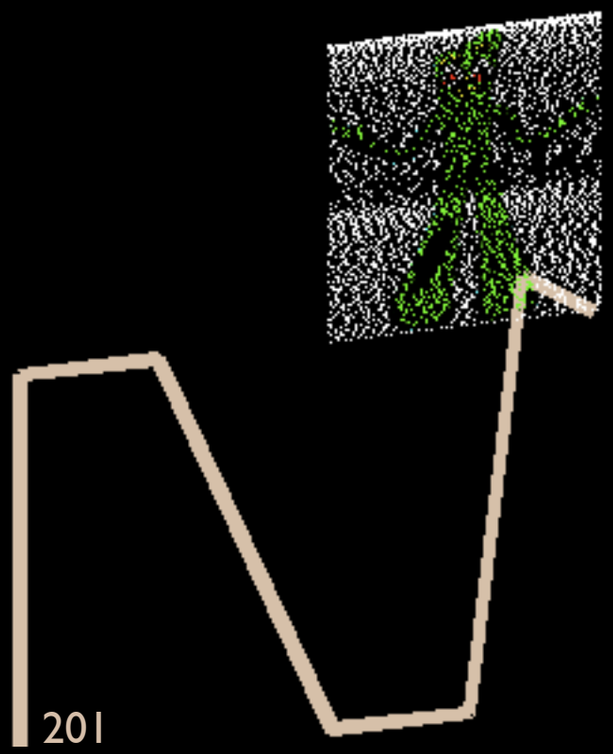
Why?

An offset to theta3 would indicate imperfect zeroing of the encoder or that the desk is tilted relative to gravity.

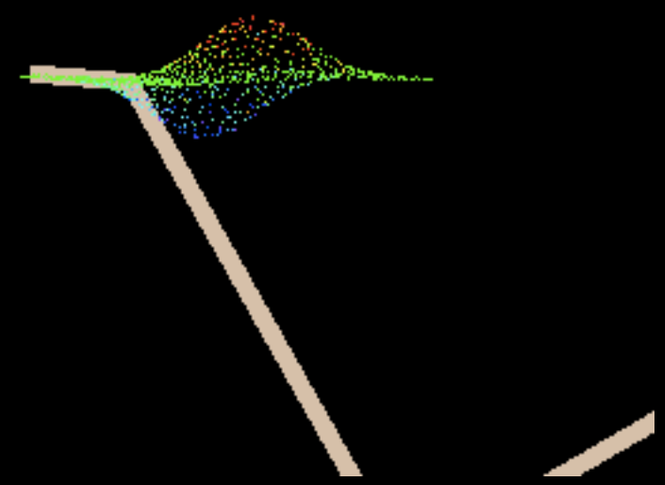
An offset to the torque is most likely a current output bias, i.e., nonzero current when we command zero.

Getting a robot to perform **brilliantly** requires that you understand all aspects of the system, from the lowest level all the way to the highest level.

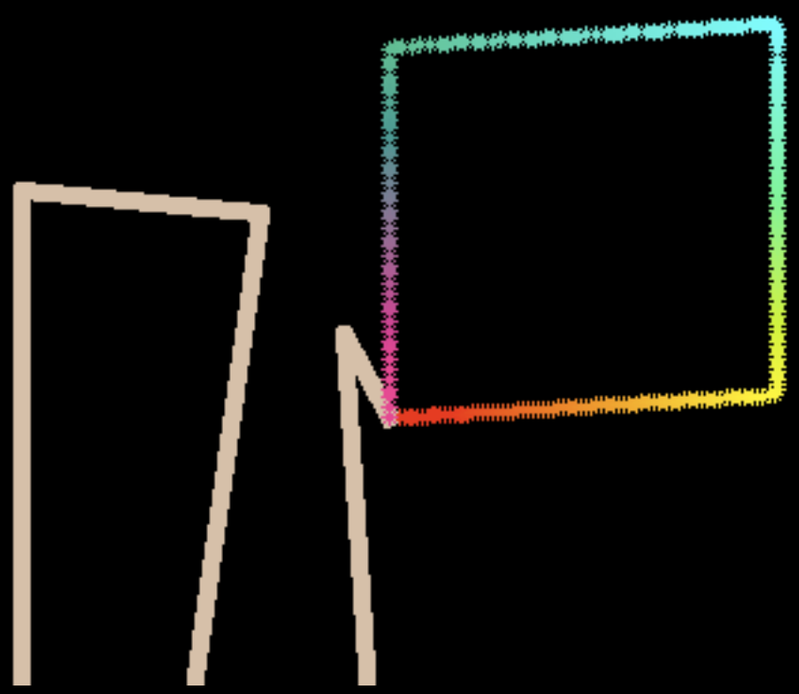
I was reminded of it last night as I worked to get the PUMA ready to film your lovely light paintings.



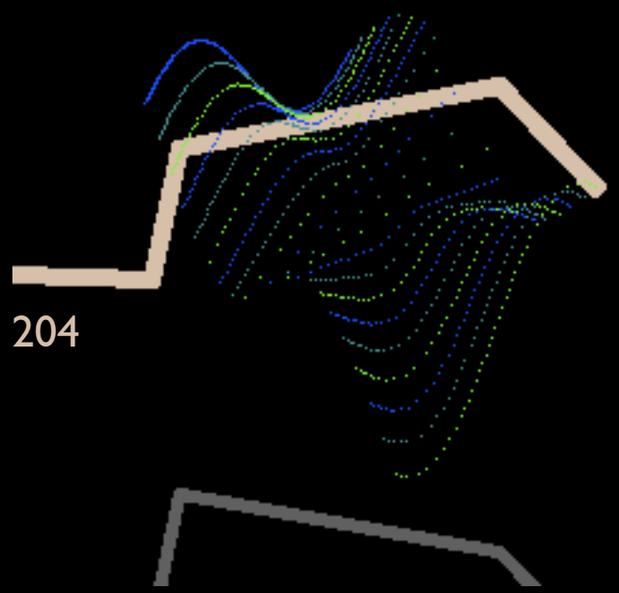
201



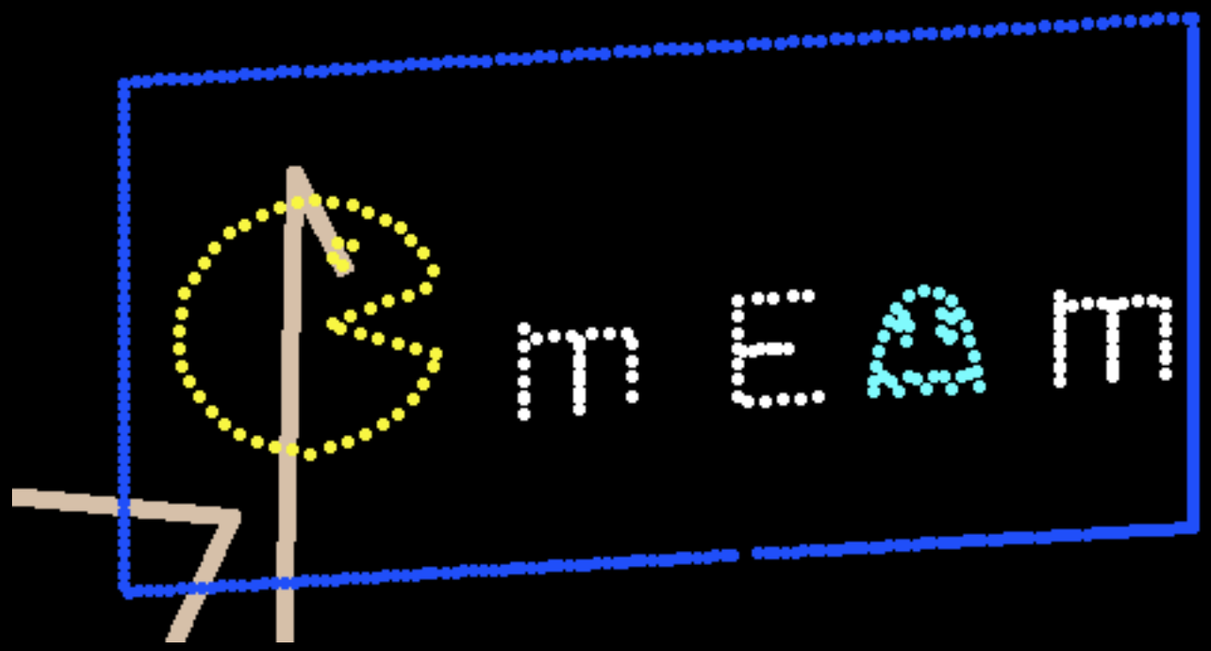
202



203



204

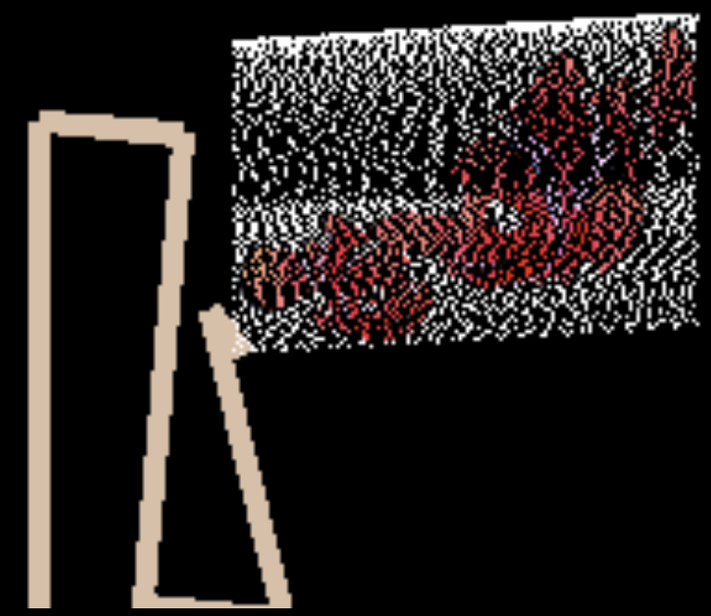


205

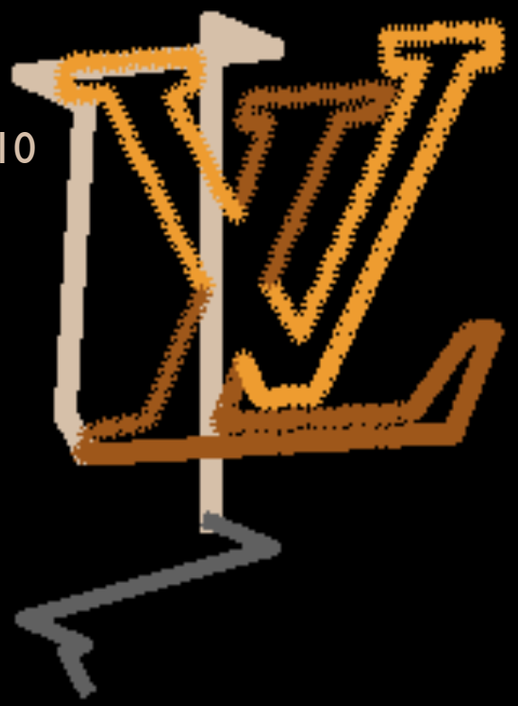
206

207

208

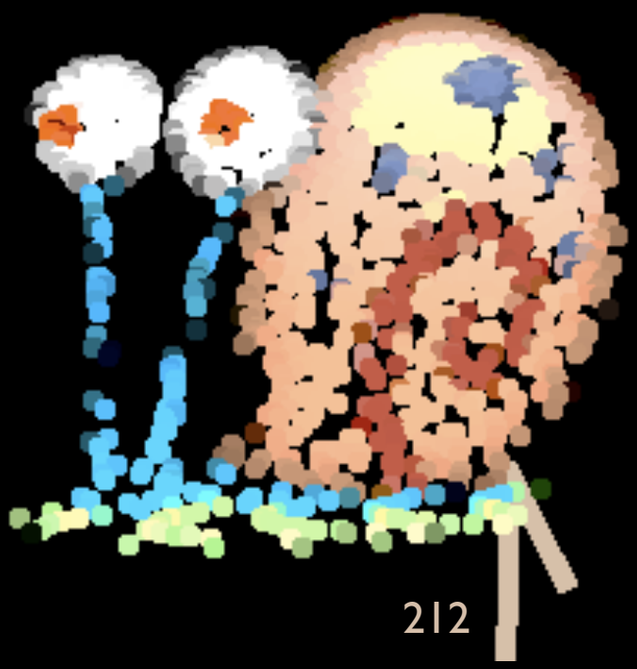


209



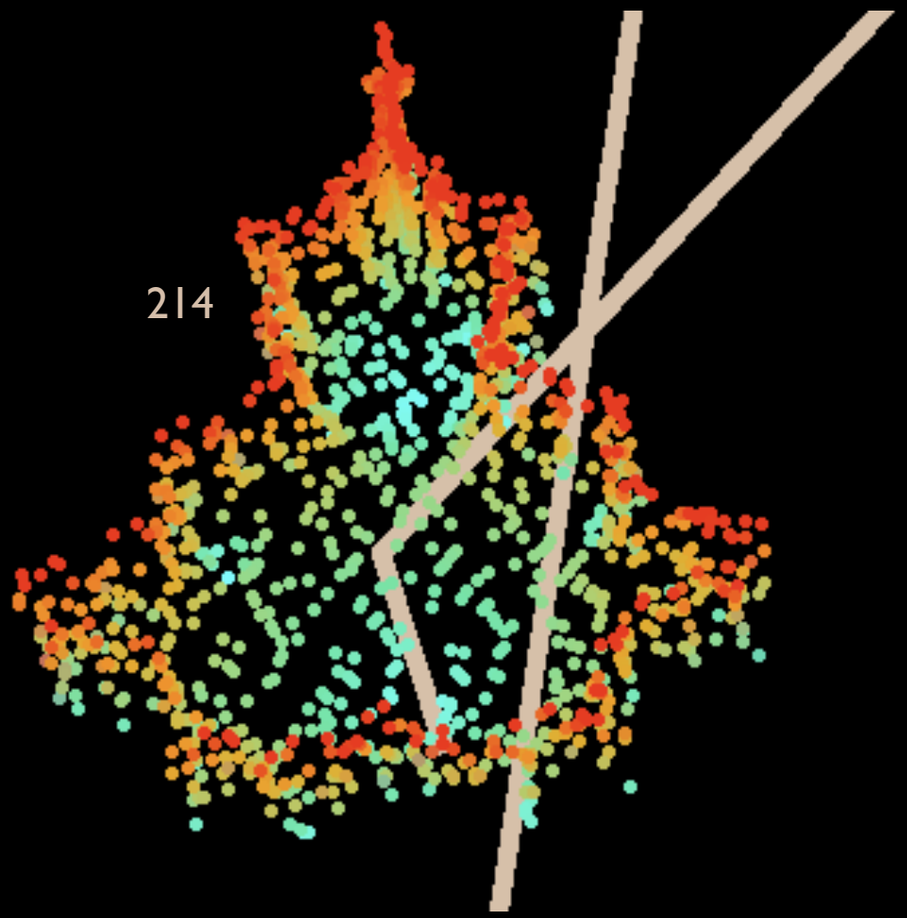
210

211



212

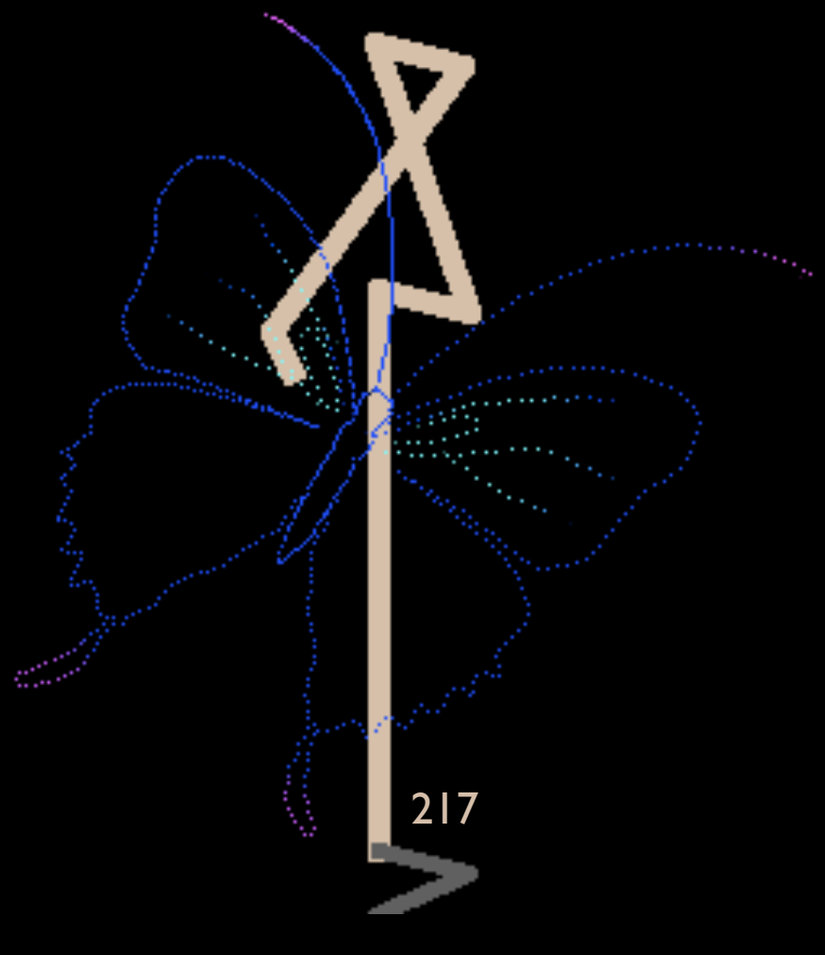
213



214

215

216

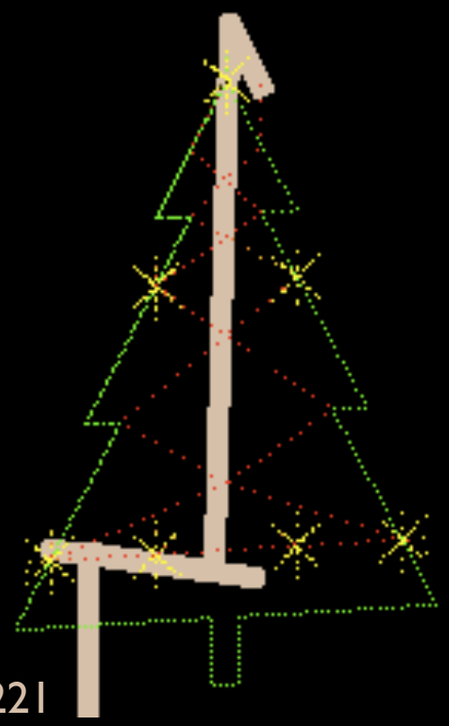


217

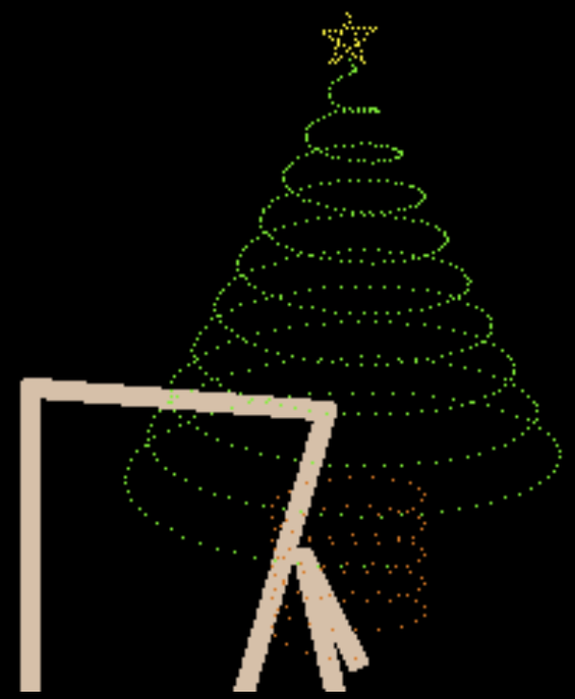
218

219

220

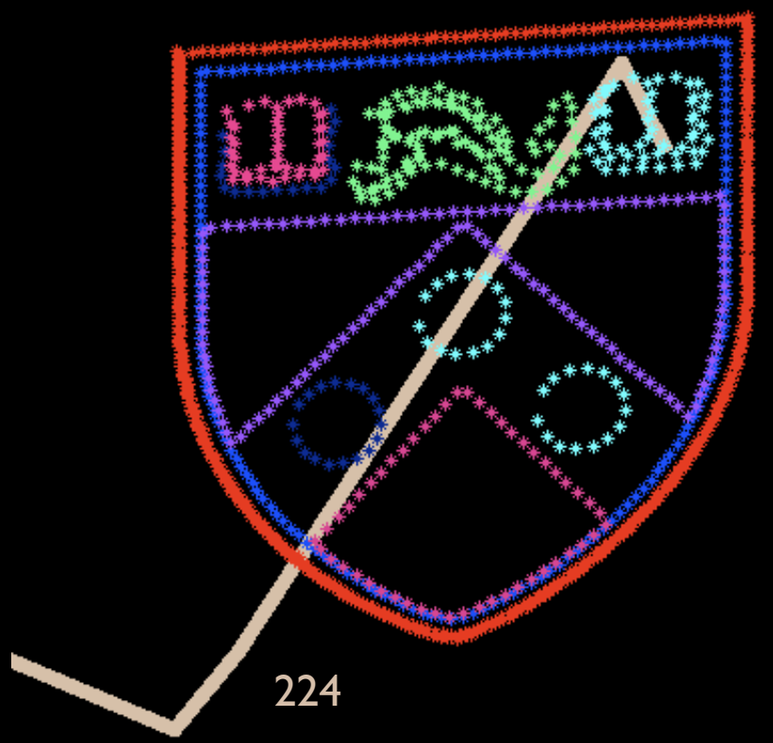


221



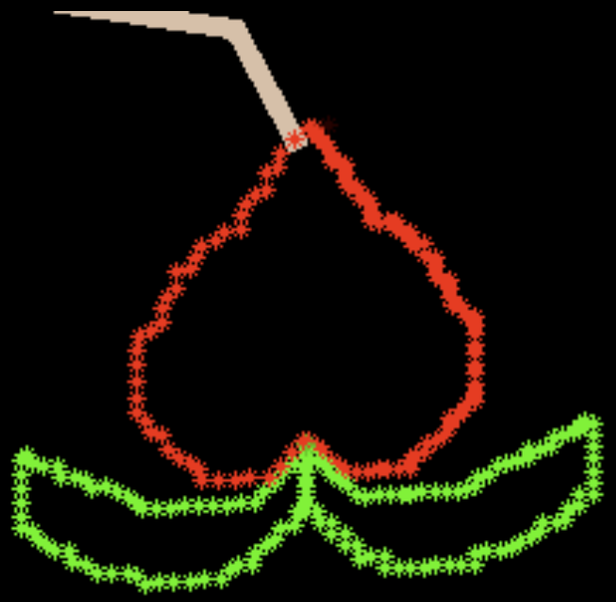
222

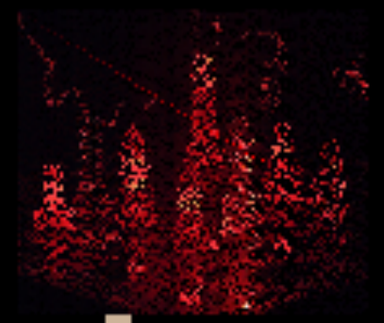
223



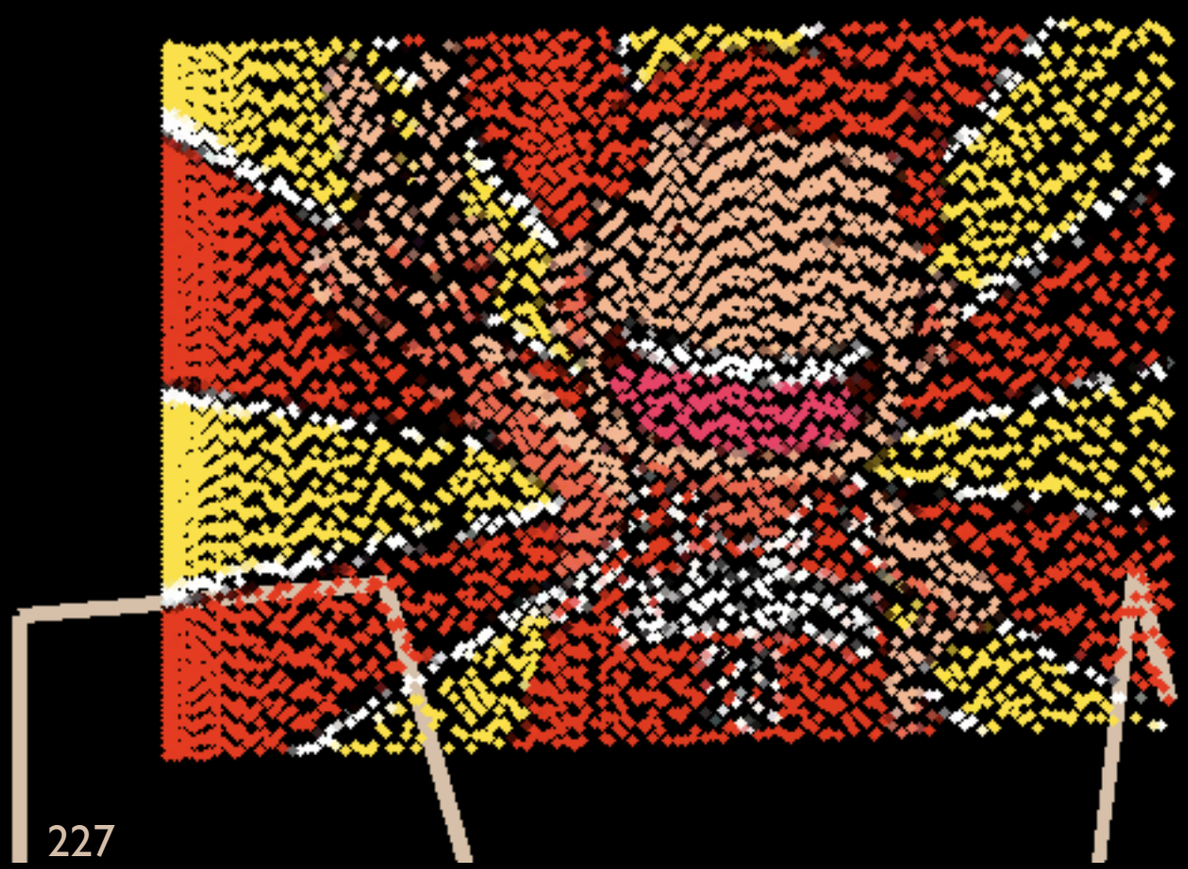
224

225





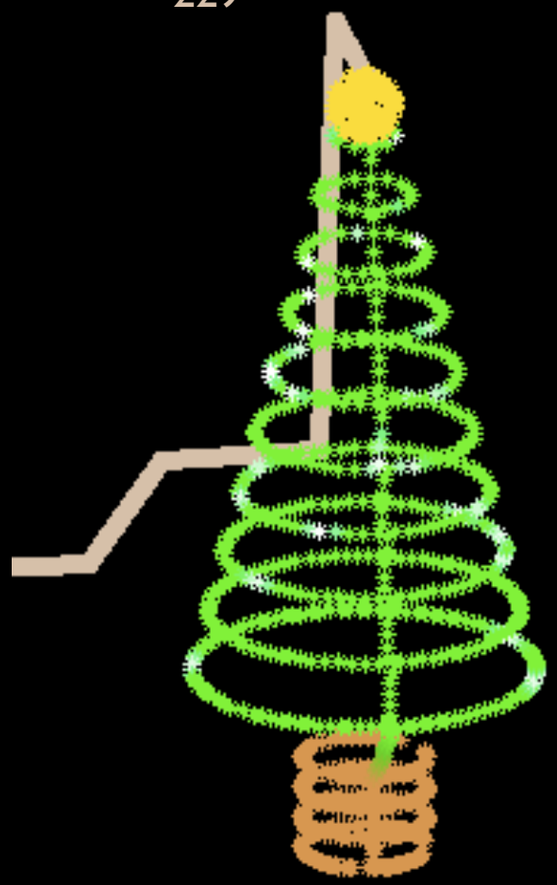
226



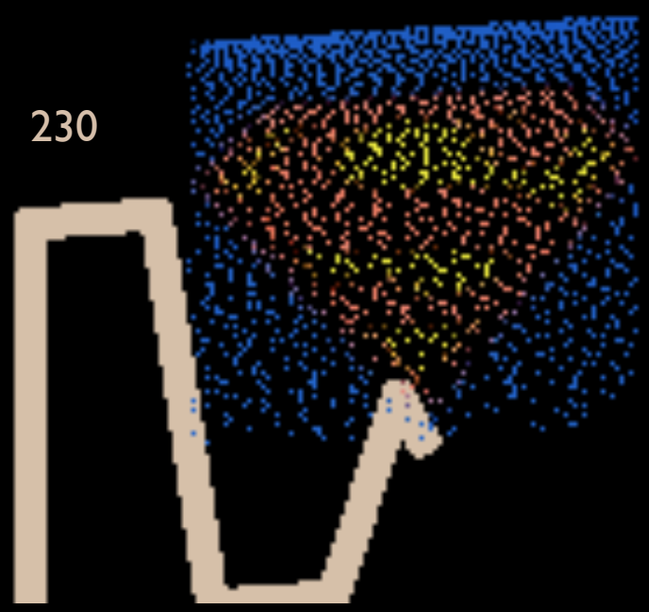
227

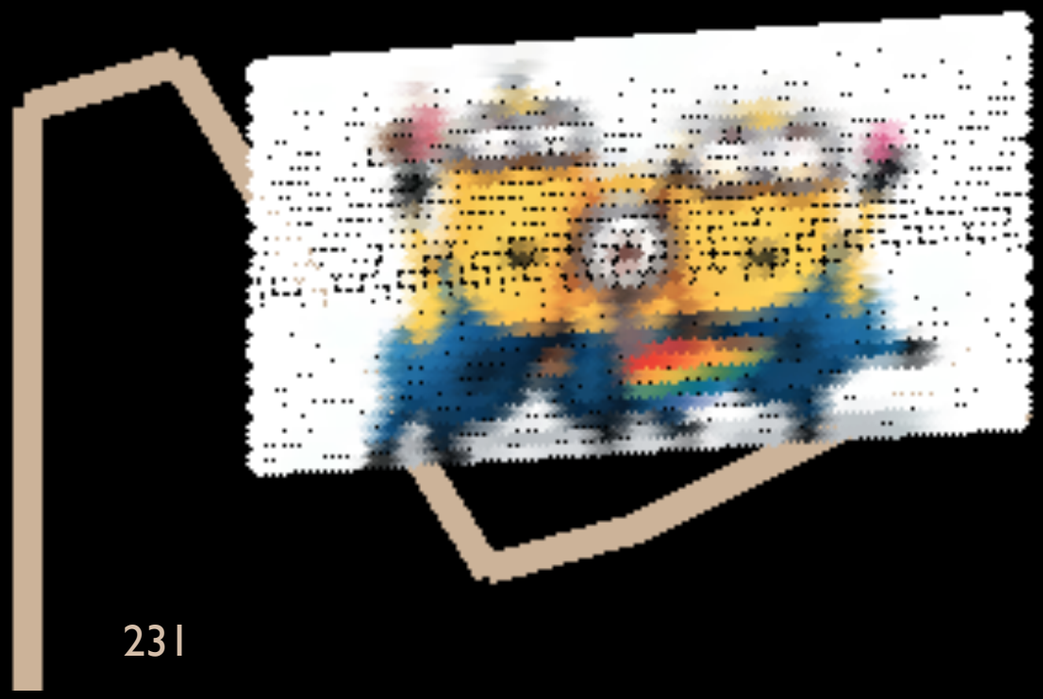
228

229



230





231

Test recording for team 210 with webcam #1

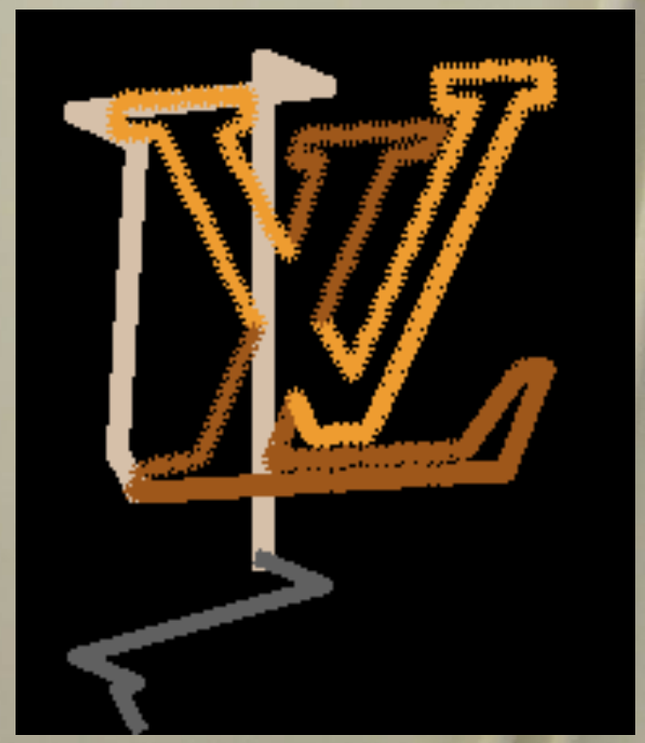
Test recording for team 210 with webcam #2

The light paintings are currently coming out over-exposed, with the colors distorted. :(

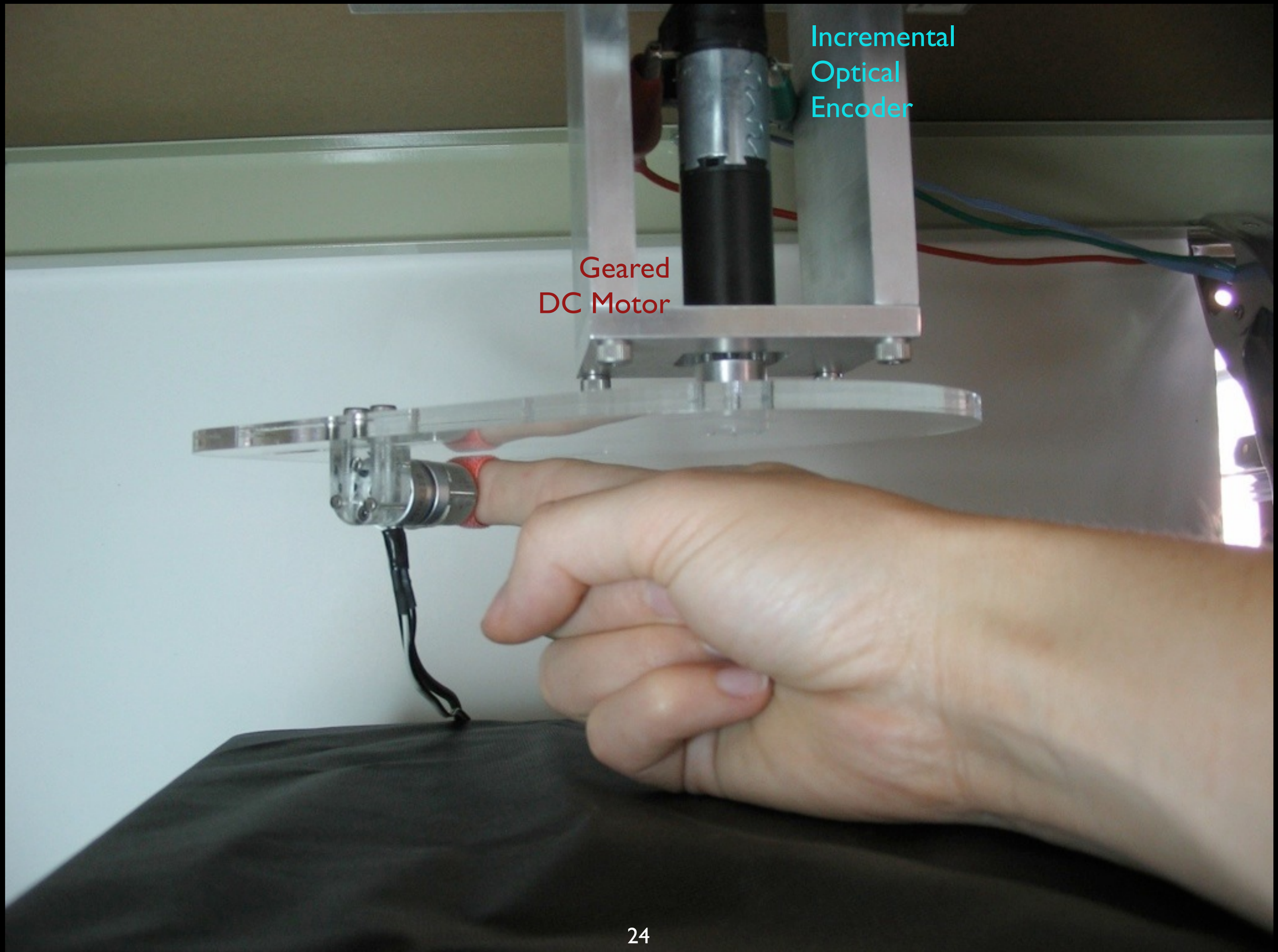
I am going to pick a different camera and update how the composite image is being made.

The LED is also flashing as the robot moves. Weird! This problem is probably electrical, maybe software.

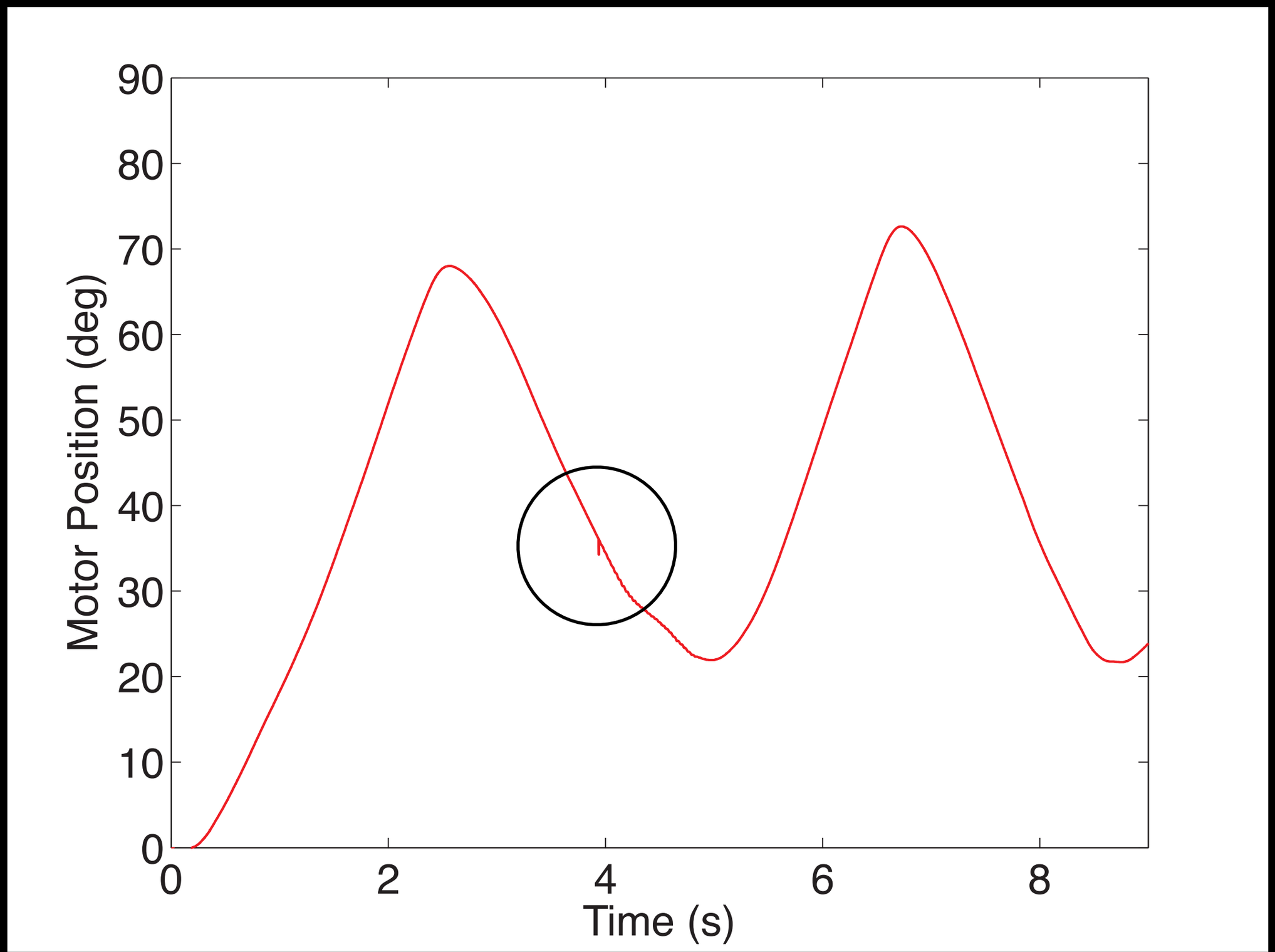
I will announce final light painting instructions as soon as they are ready.



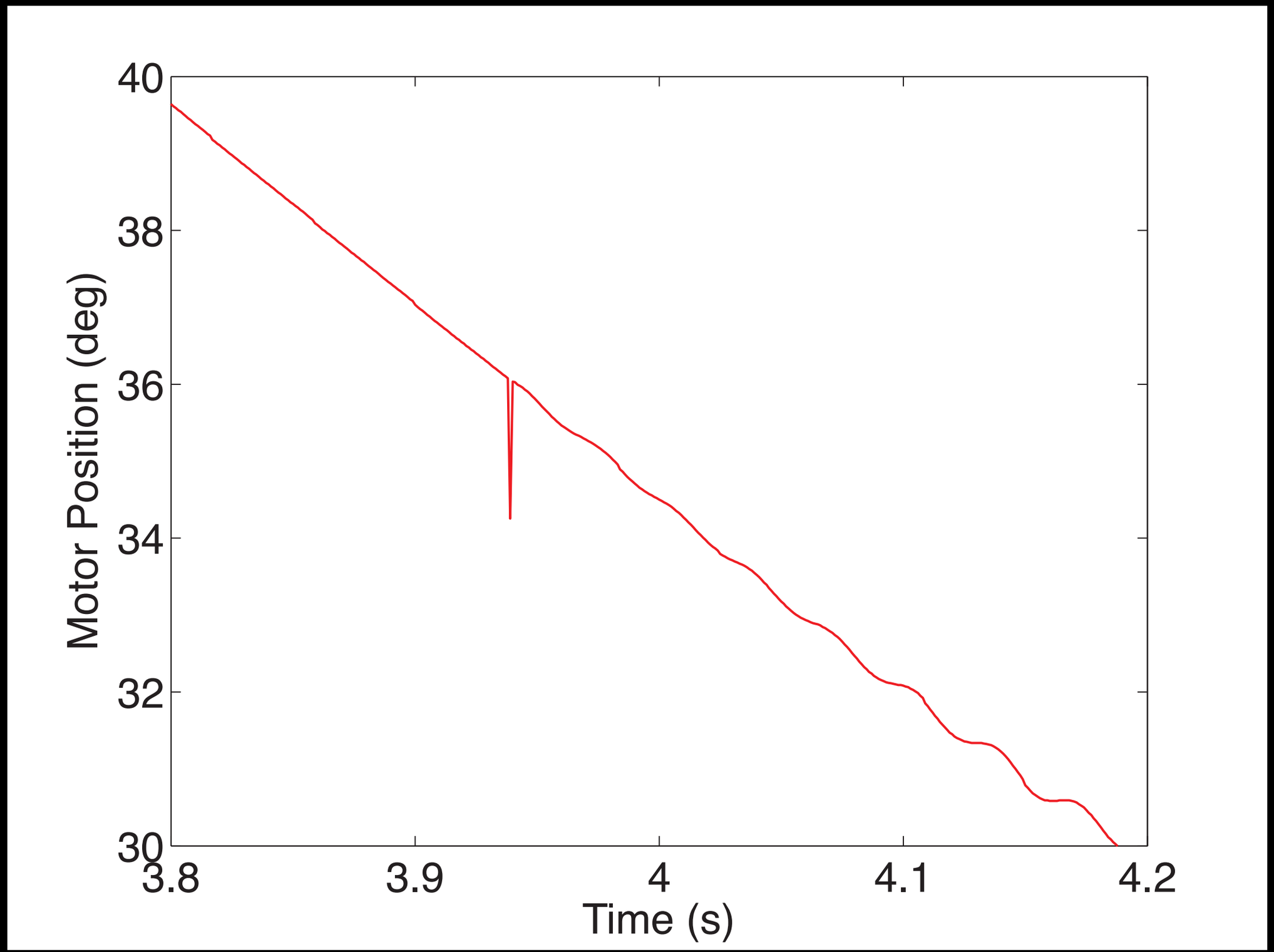
A sample custom haptic device for index finger flexion and extension


Incremental
Optical
Encoder
Geared
DC Motor

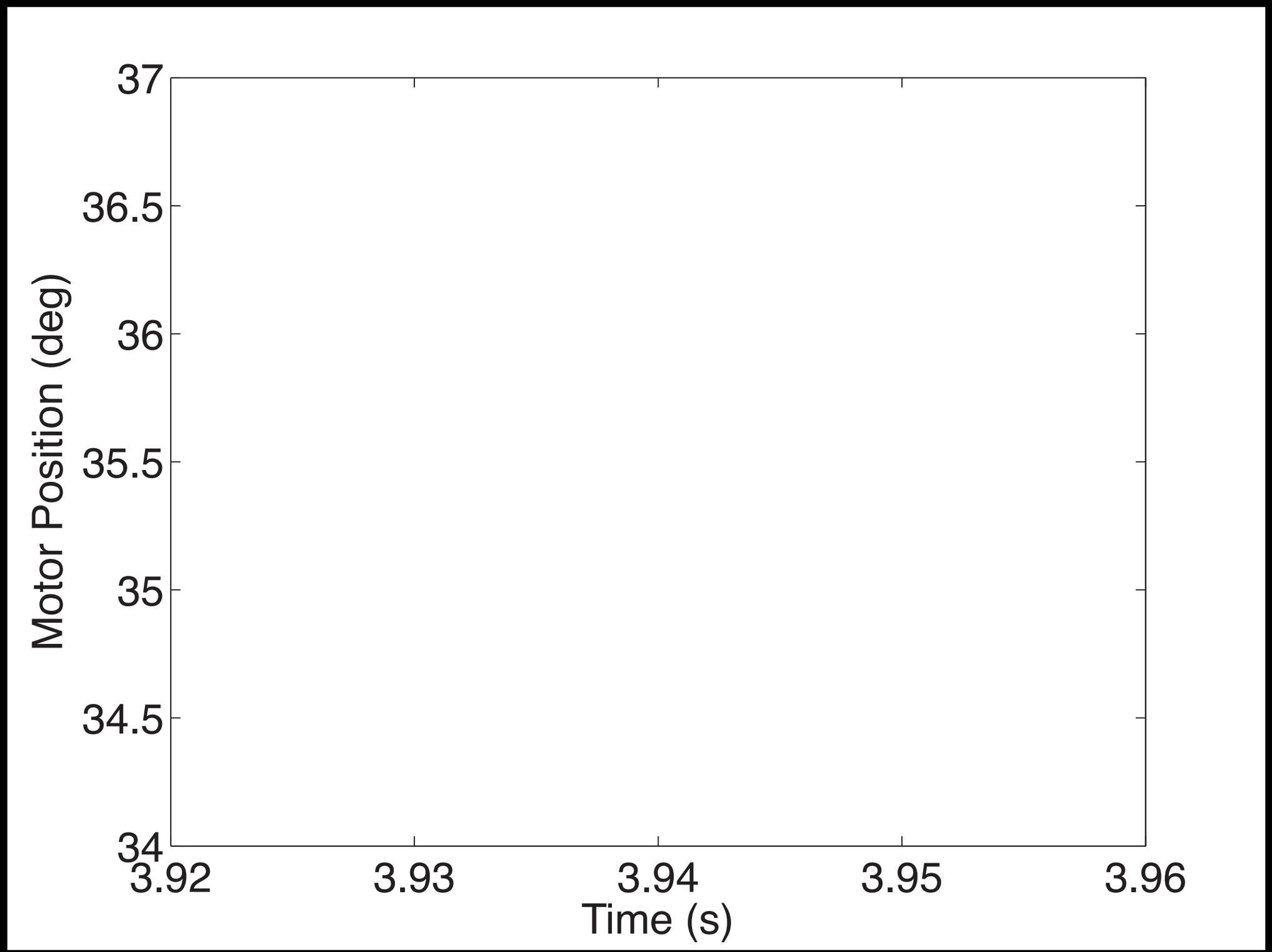
$$\tau_m = k_p(\theta_d - \theta_m) + k_d(\omega_d - \omega_m)$$


90
80
70
60
50
40
30
20
10
0
Motor Position (deg)
0
2
4
6
8
Time (s)

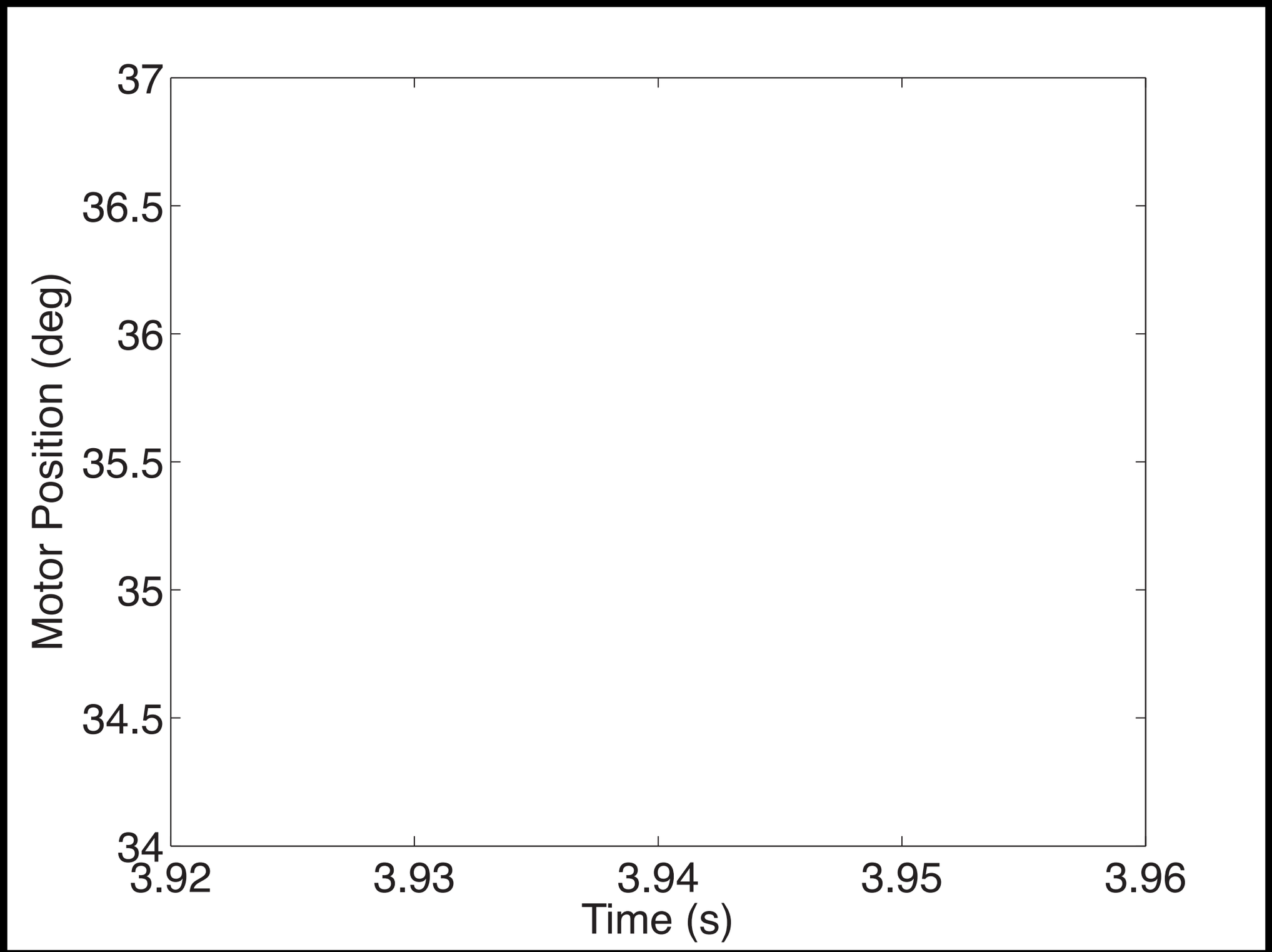
What's happening?



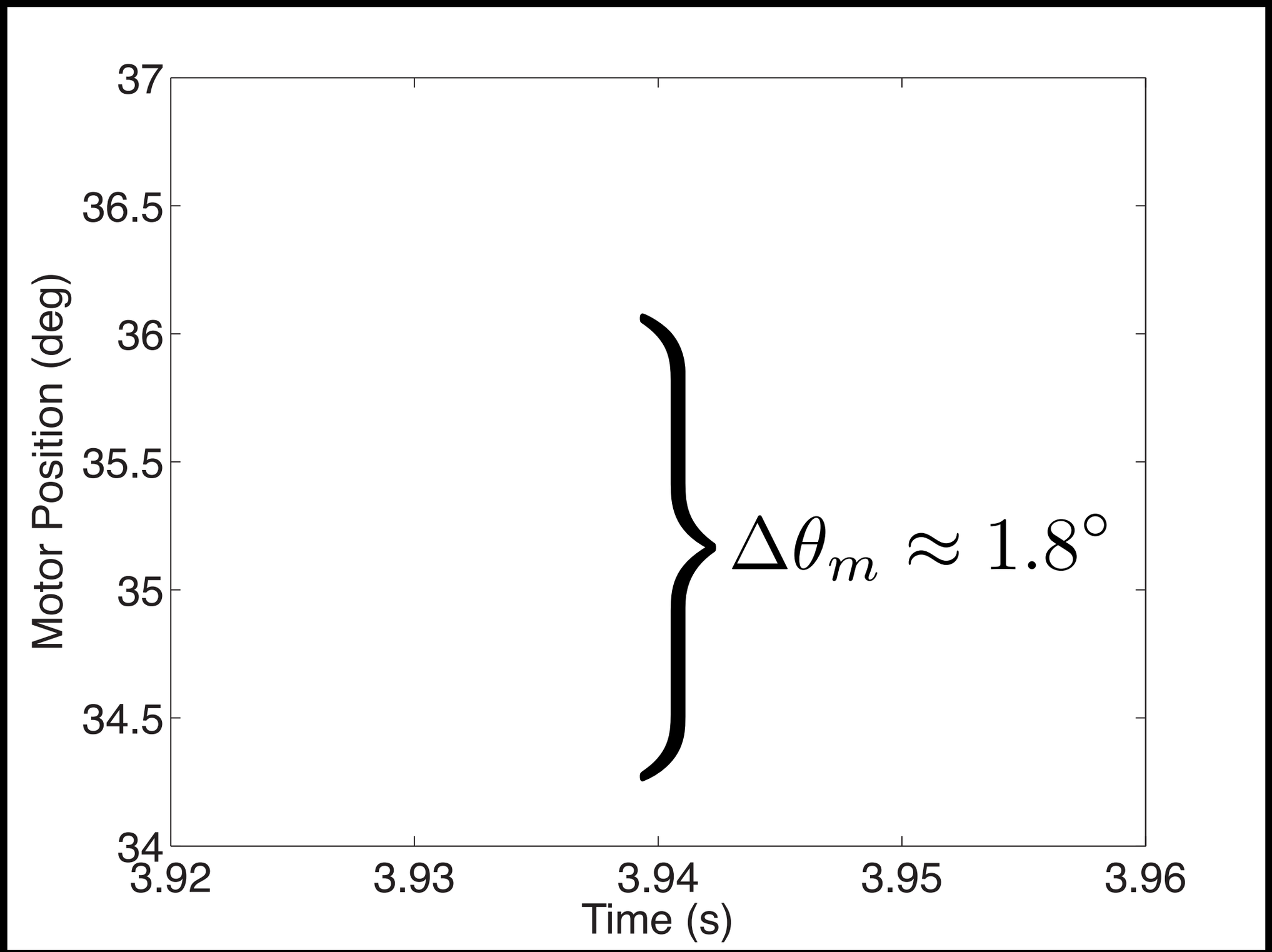
One data point is out of line.



That data point should be about where the black dot is.



Be a robotics detective. What could you check?



$$\Delta\theta_m = 1.8^\circ$$

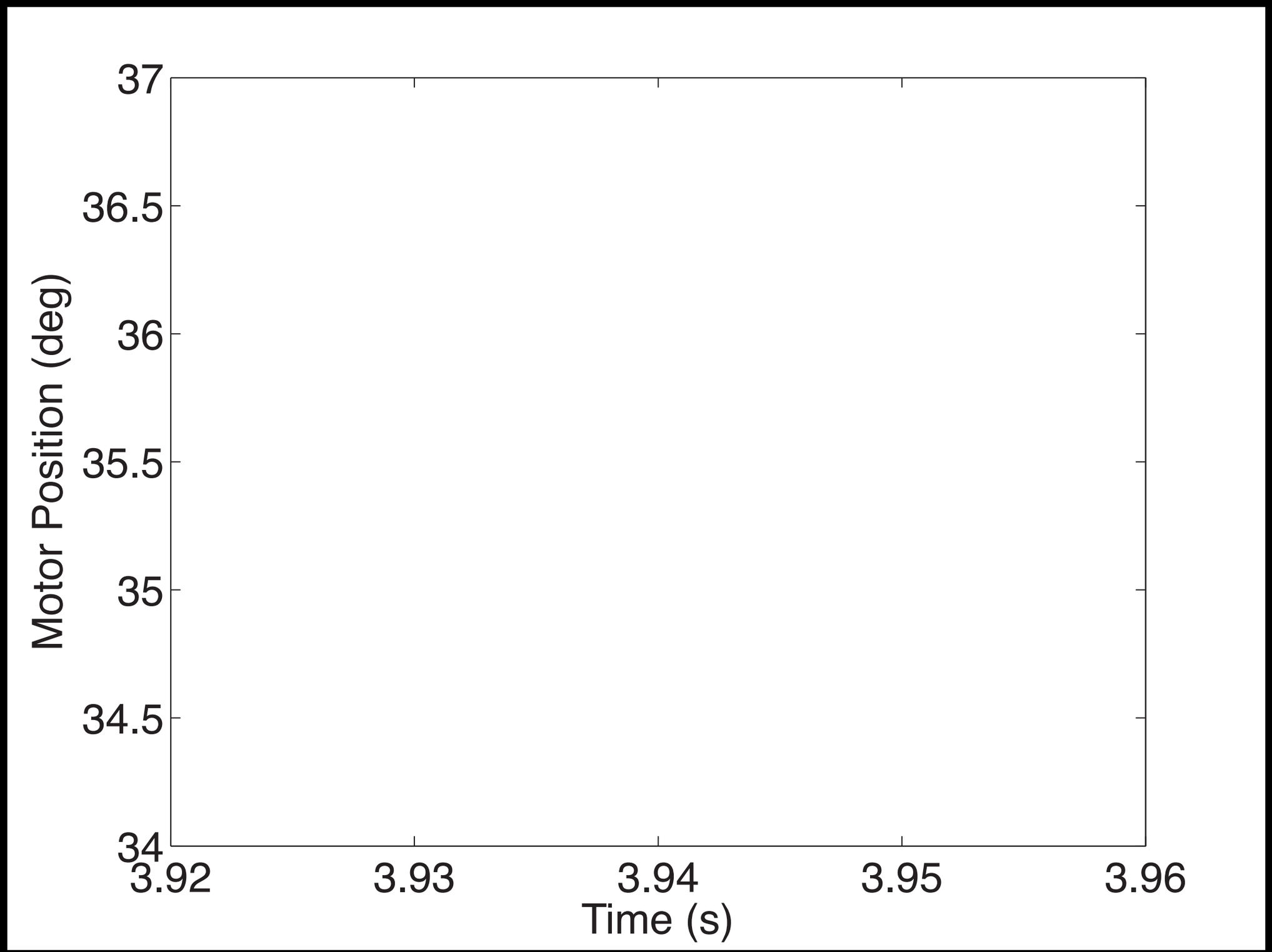
$$\Delta\theta_m = 1.8^\circ \cdot \frac{51200 \text{ counts}}{360^\circ}$$

$$\Delta\theta_m = 256 \text{ counts}$$

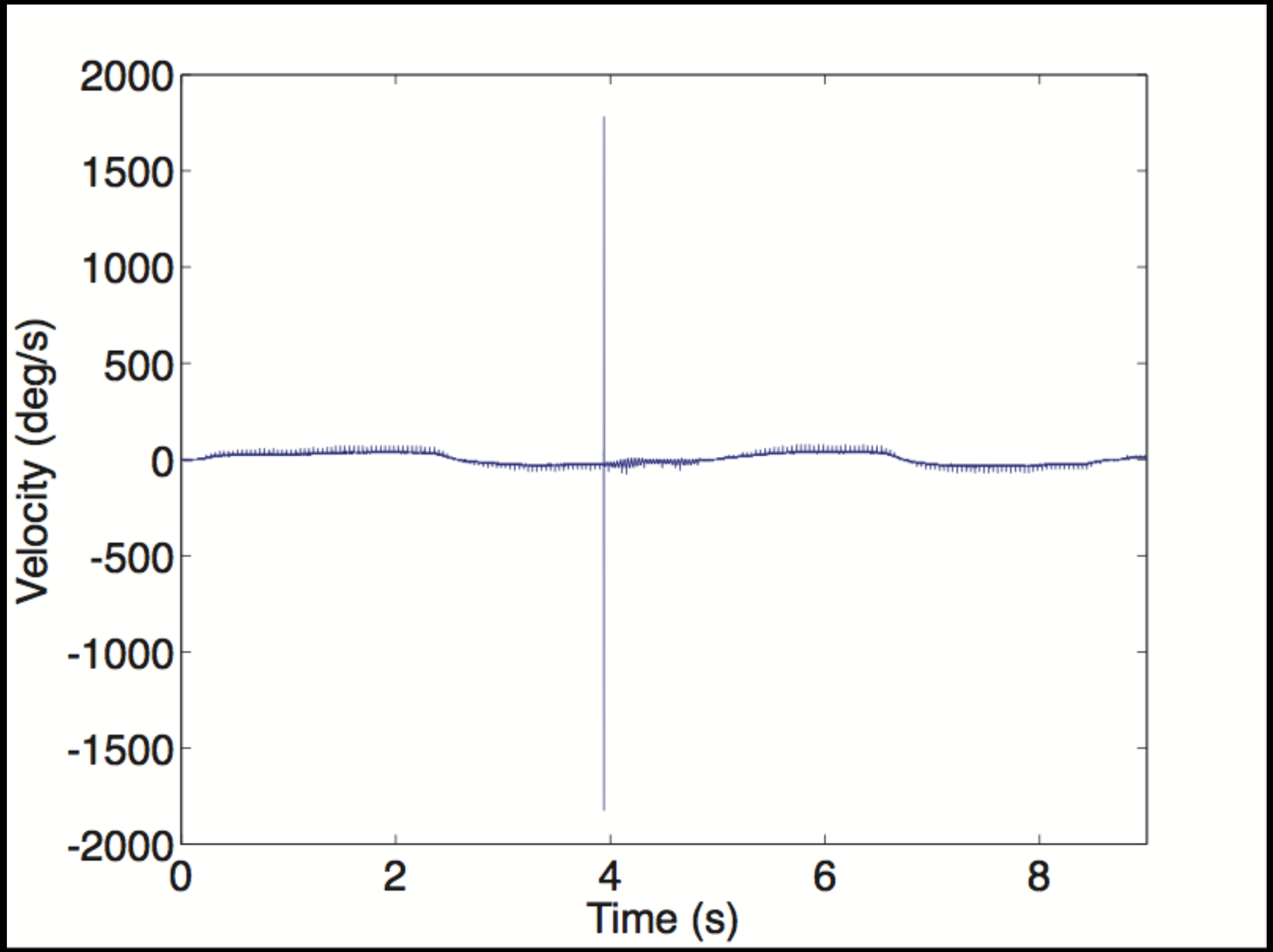
$$\Delta\theta_m = 2^8 \text{ counts}$$

The eighth bit of the count was occasionally flipping from 0 to 1 or vice versa

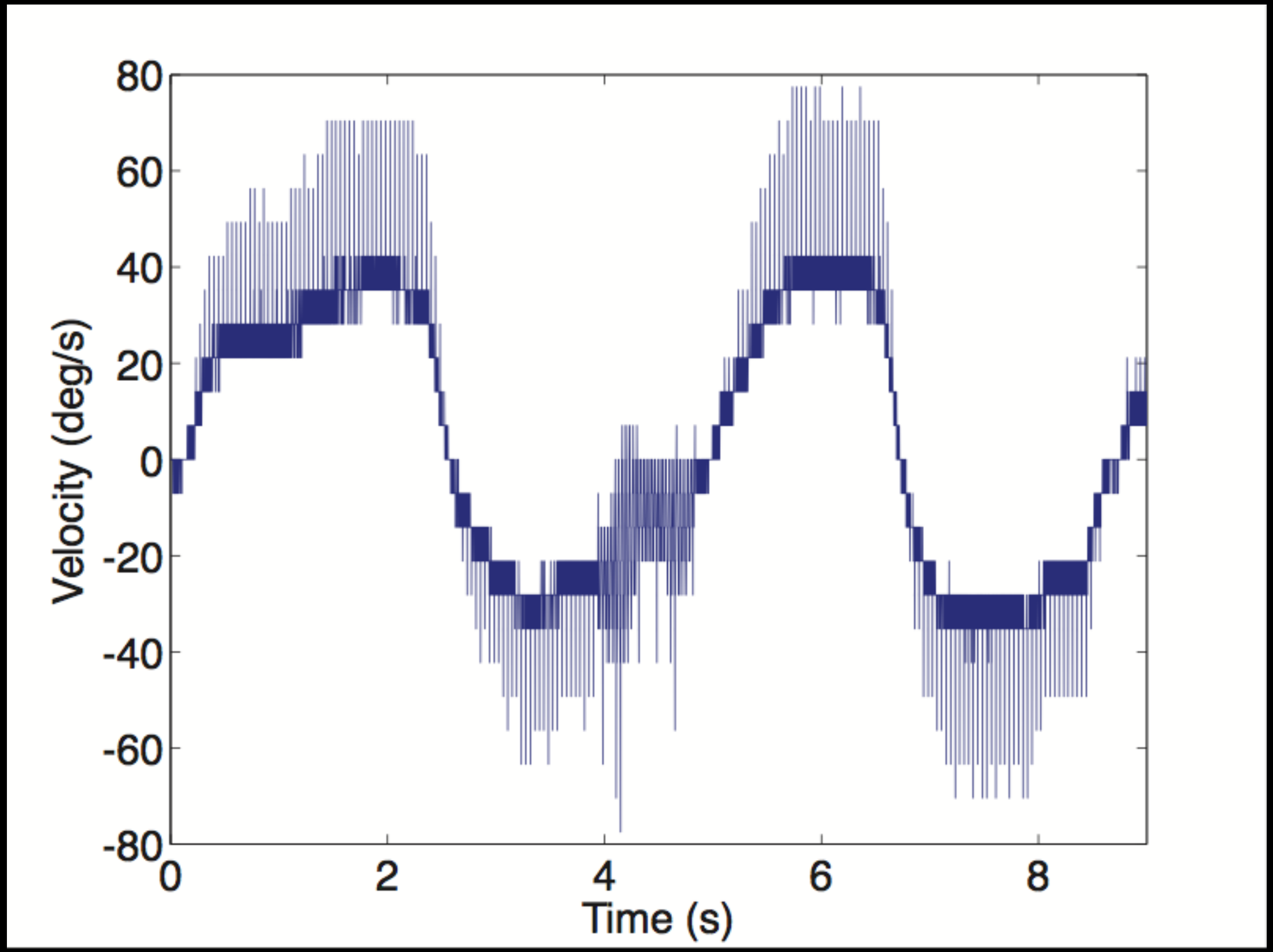
Fix glitch by checking magnitude of change and fixing bit flip.



Previous Motor Velocity Plot

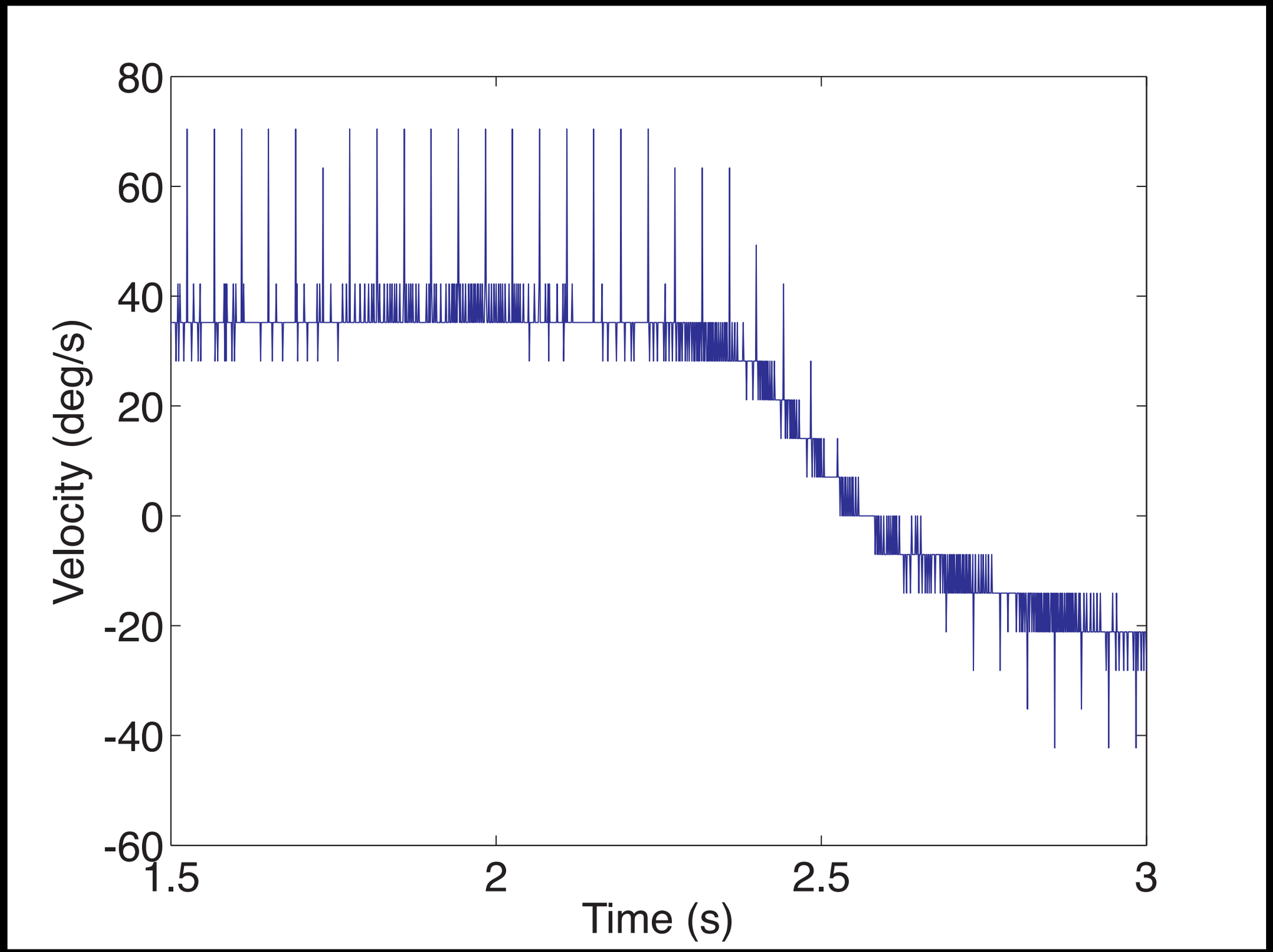


New Motor Velocity Plot

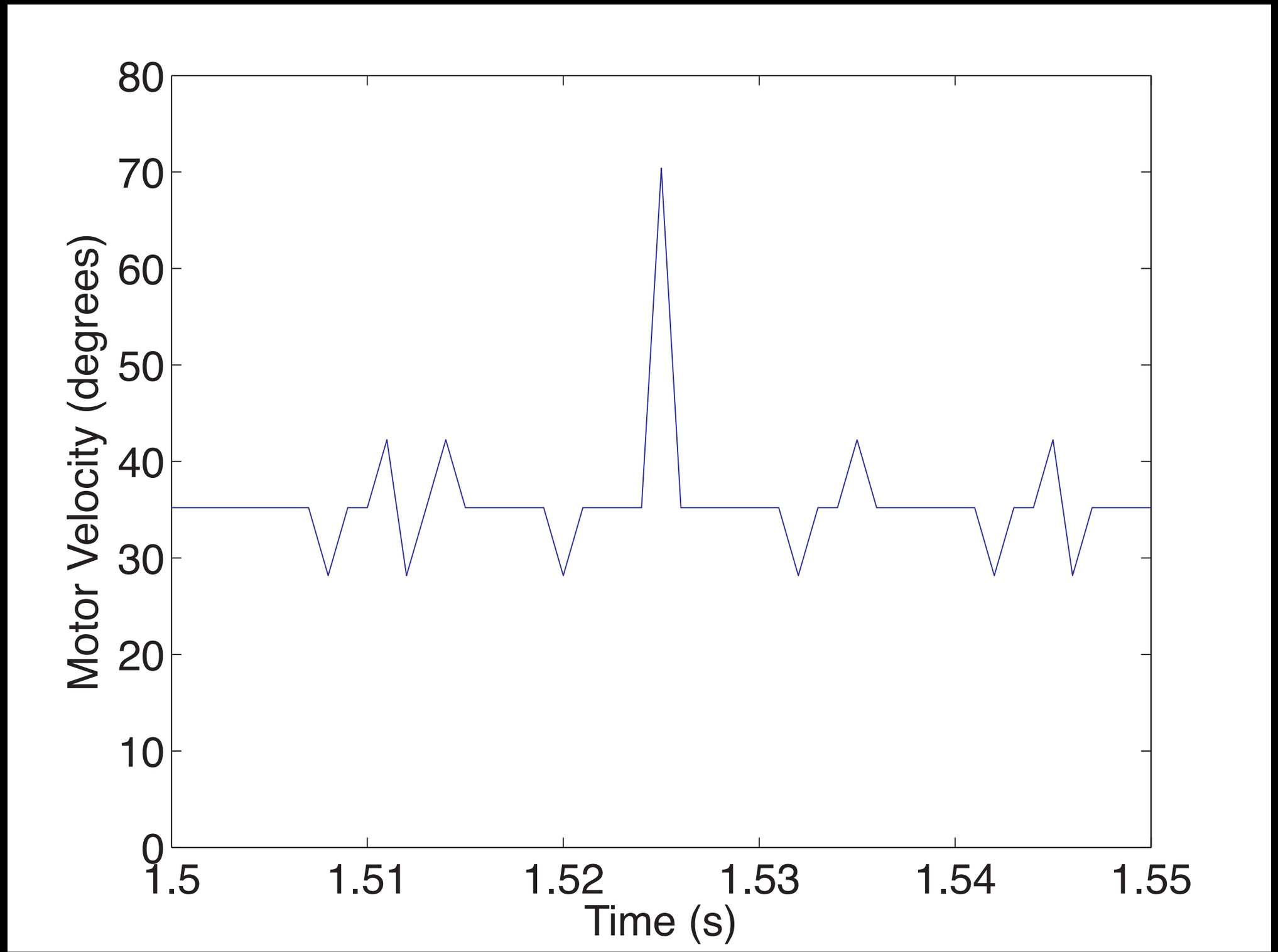


What should I do to this signal?

Be suspicious of unexpected patterns. Are the spikes noise?

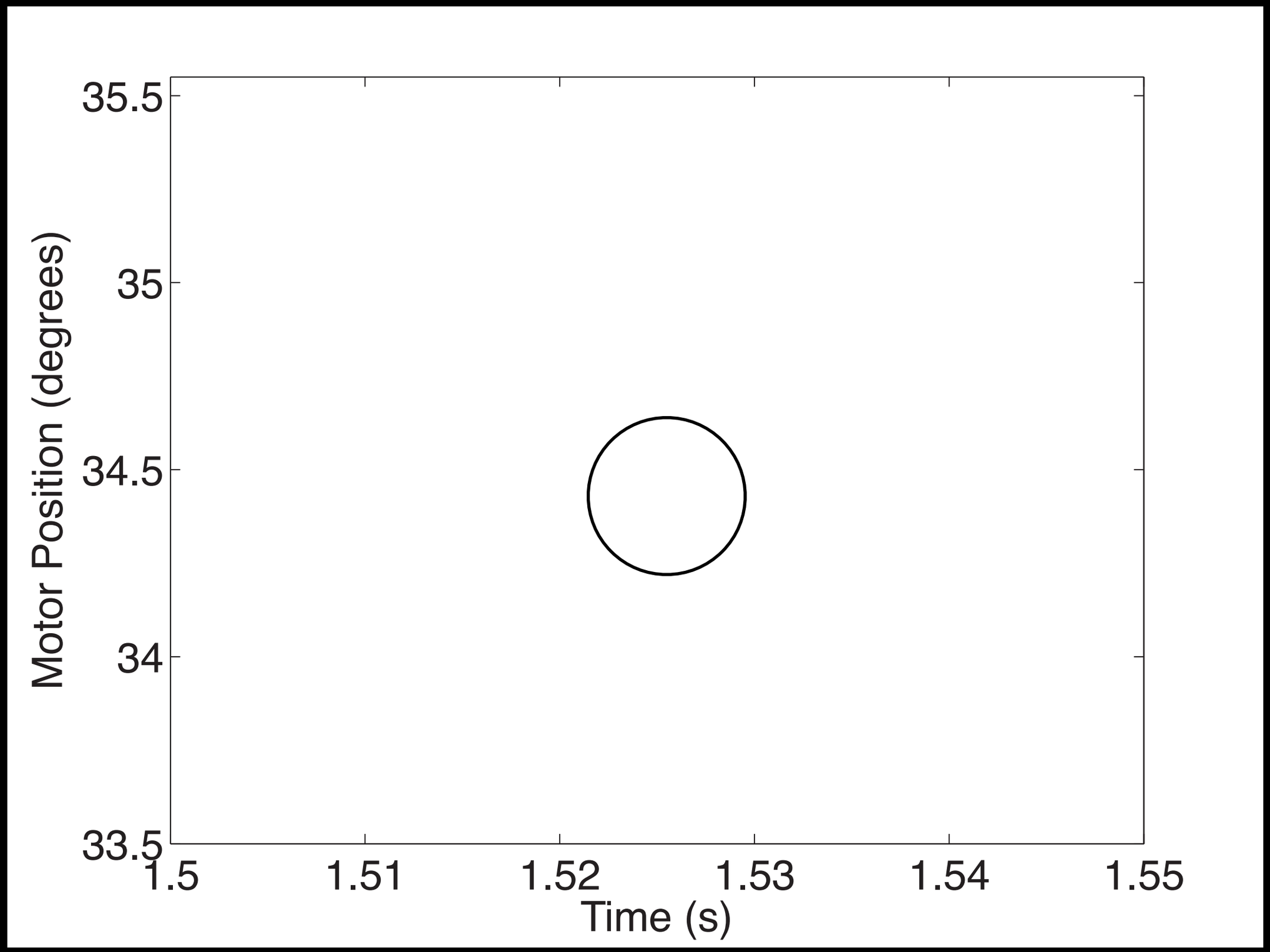


What do you notice about this spike?



The spike magnitude is twice as large as the average signal.

Plot individual position measurements as dots, not a line.

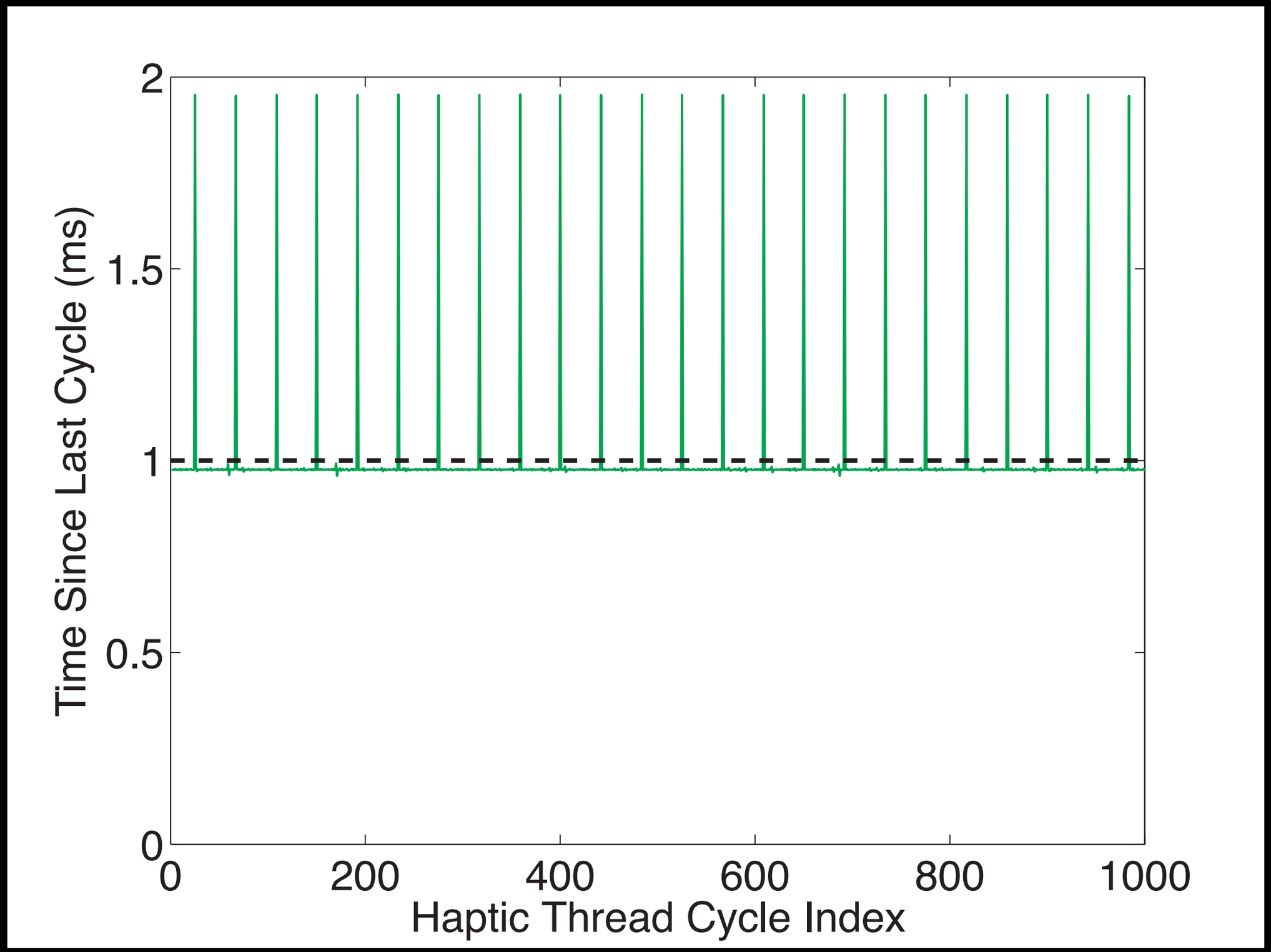


Be suspicious of all aspects of your system. Question assumptions.

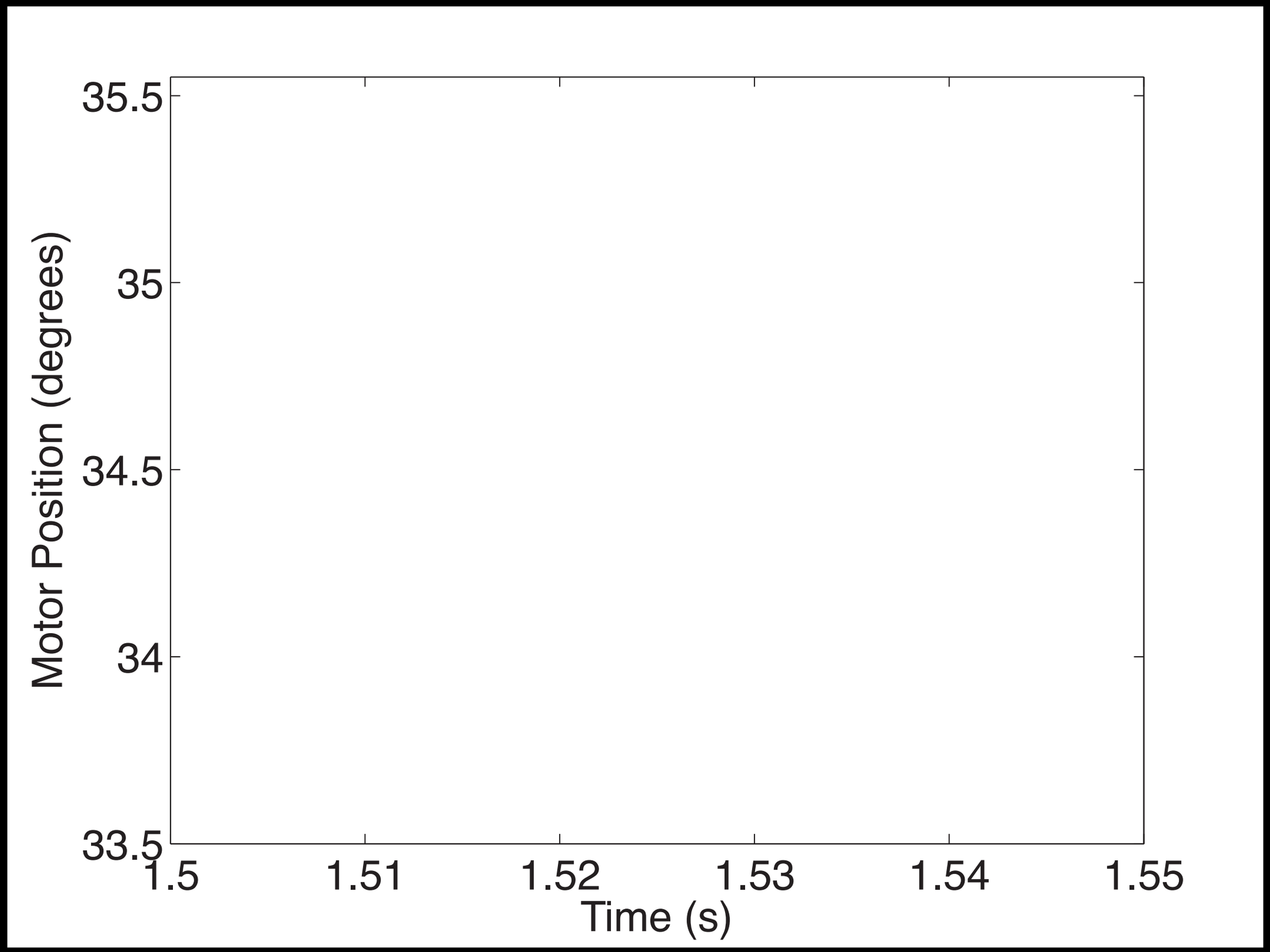
```
/*****************************************************************
 Haptic_Function
        This is the function that updates the system's forces
*****************************************************************/


void __stdcall Haptic_Function(void *pv)

{
        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] = LowPass1((double)1.0/(2.0*PI*50.0), deltaTimeS, (double)tempRawVol
tage[i],  (double)filteredRawVoltage[i]);
        }

        // Handle initialization of force/torque sensor
        if ((force_bias_initialize) && (filter_wait > 50))
        {
                if (Number_of_Samples < MAX_NUMBER_OF_SAMPLES) {
                        for (int CONV_r = 0; CONV_r < 7; CONV_r++) {
                                VoltageBiasTemp[CONV_r][Number_of_Samples] = filteredRawVoltage[CONV_r];
                        }
                        Number_of_Samples++;
--(DOS)--  knob_07_01_05.cpp   63% L918    (C++ Abbrev)----------------------------------------------------------
```

```
        // *** MOTOR CONTROL ***

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

        // Read in encoder signals from the QUAD04 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 || fabs(FTValues[1])>200 || fabs(FTValues[2])>500 || fabs(FTValues[3])>15
00 || fabs(FTValues[4])>1500 || 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 sta
tes.
                switch (state) {
                case waitingForParameters:
                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;
```

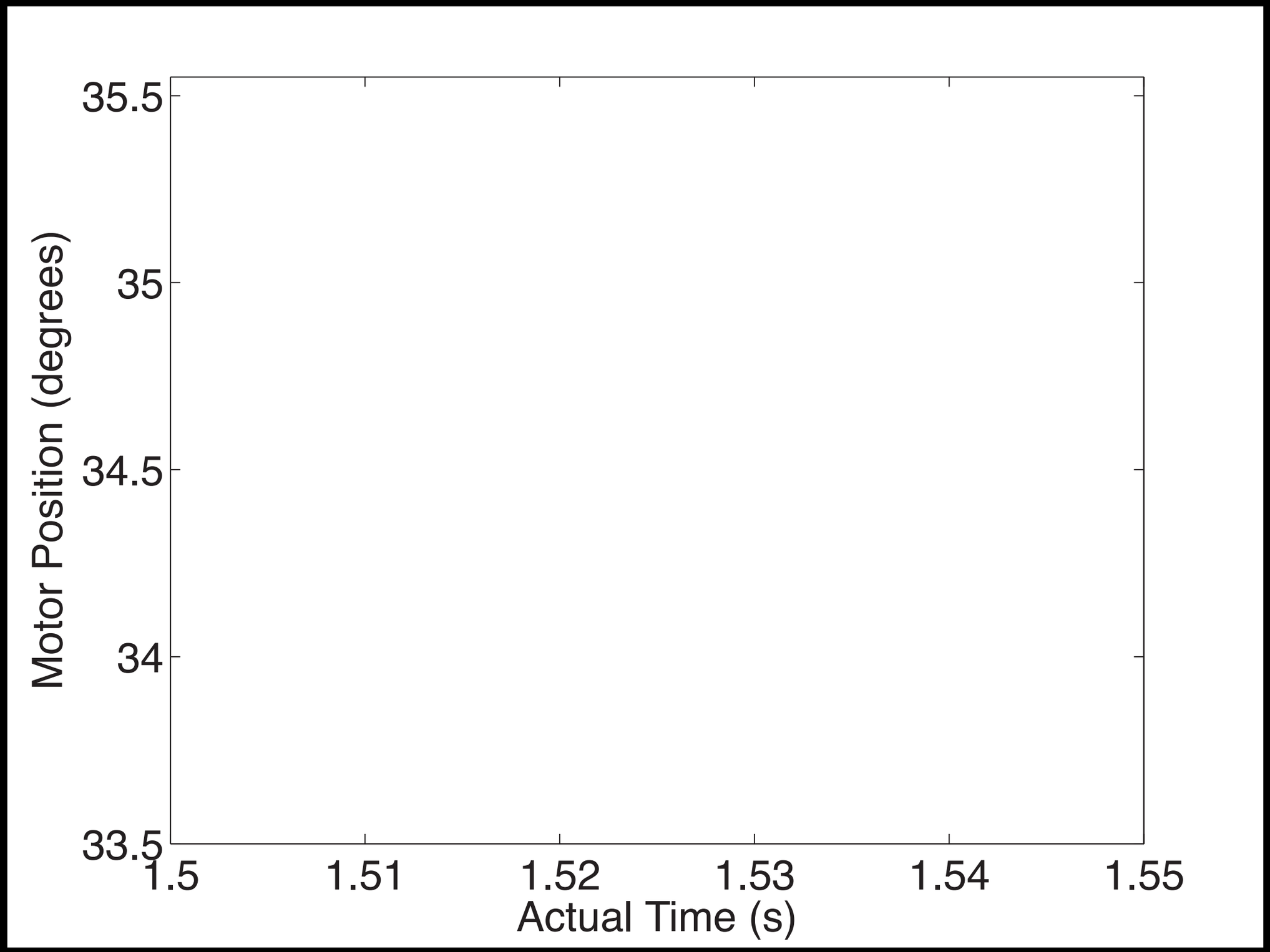
My code was not executing at an even time interval.



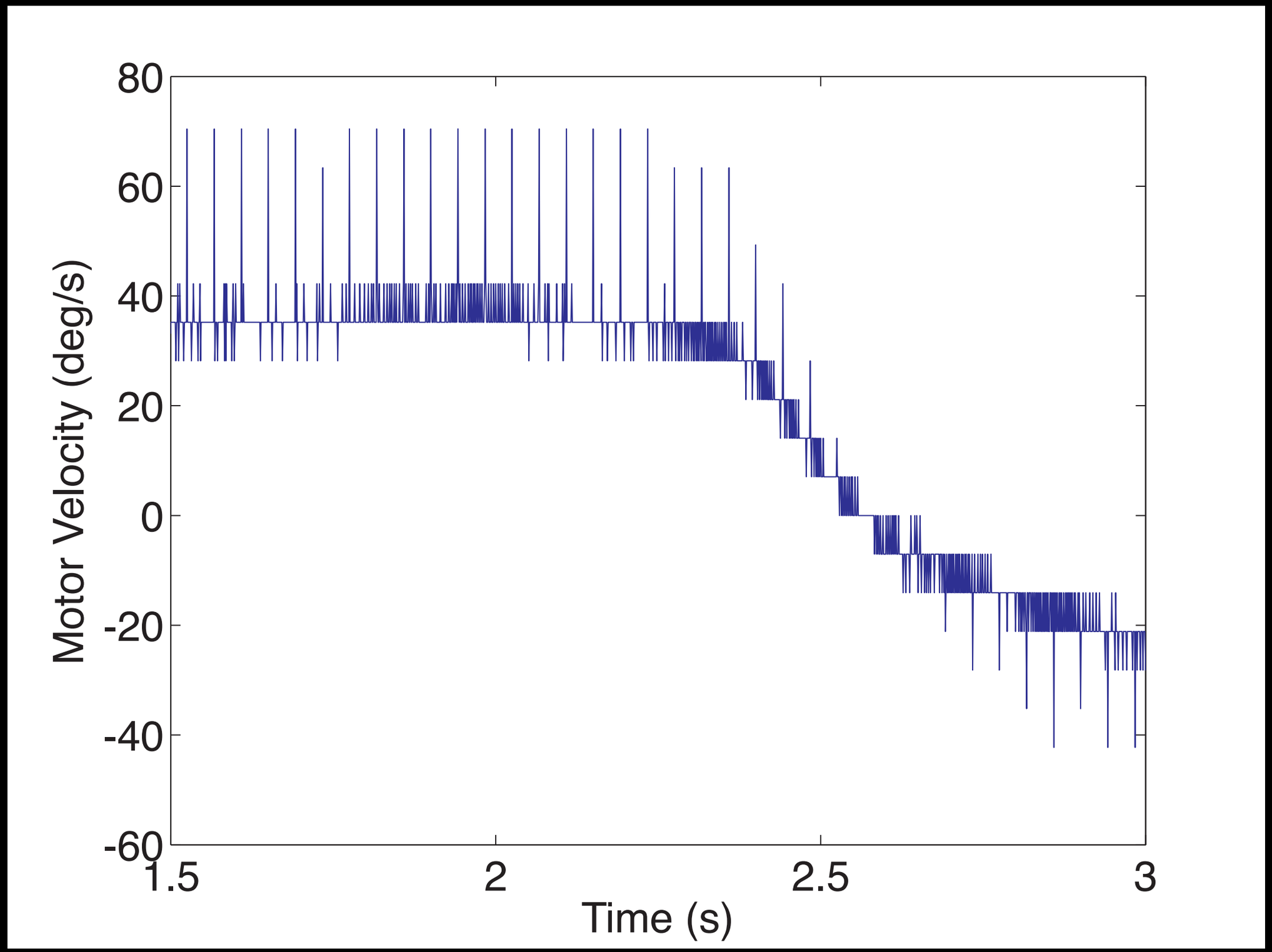
Position versus Assumed Time



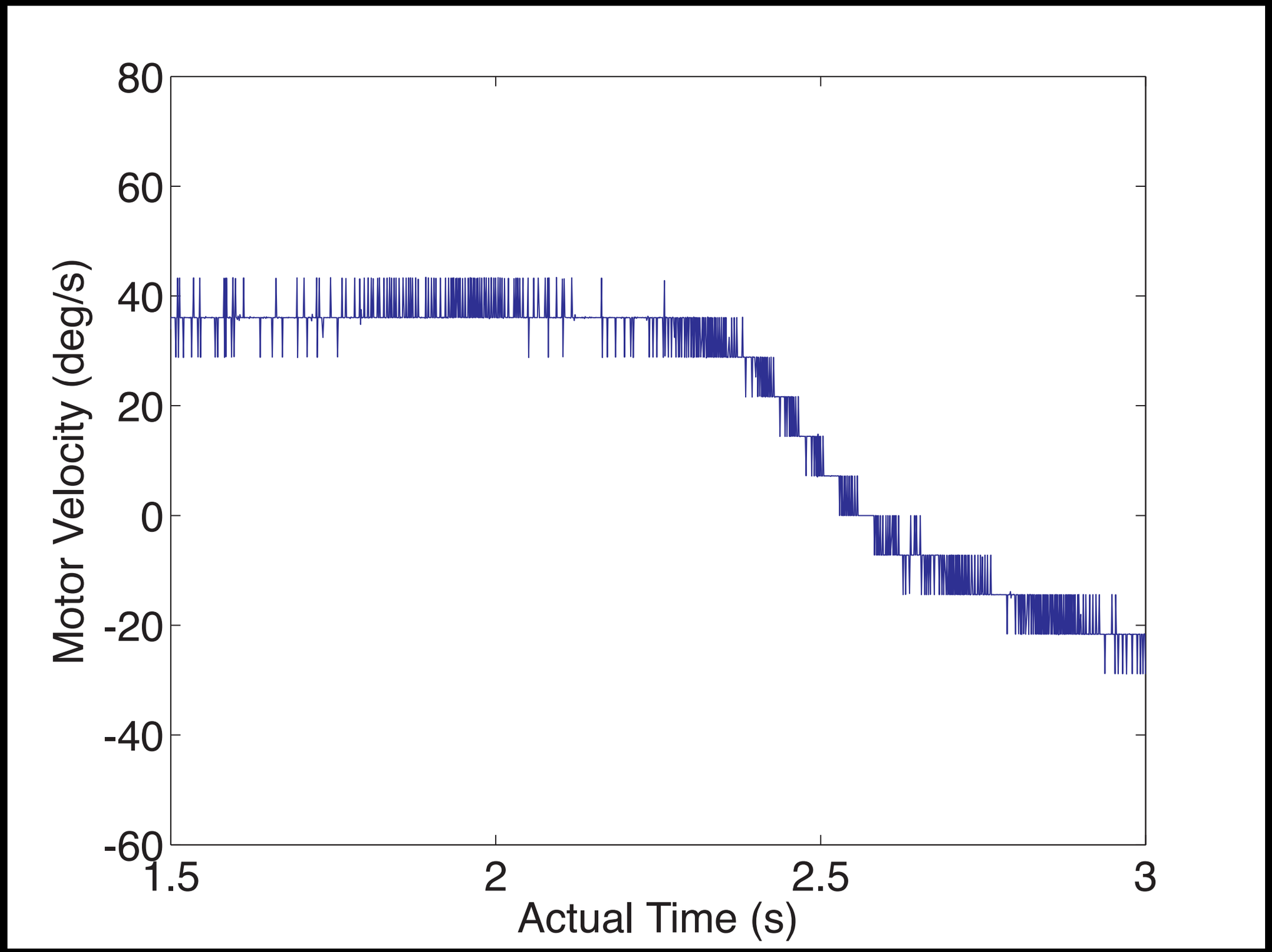
Position versus Actual Time



Motor Velocity Plot using Assumed Time



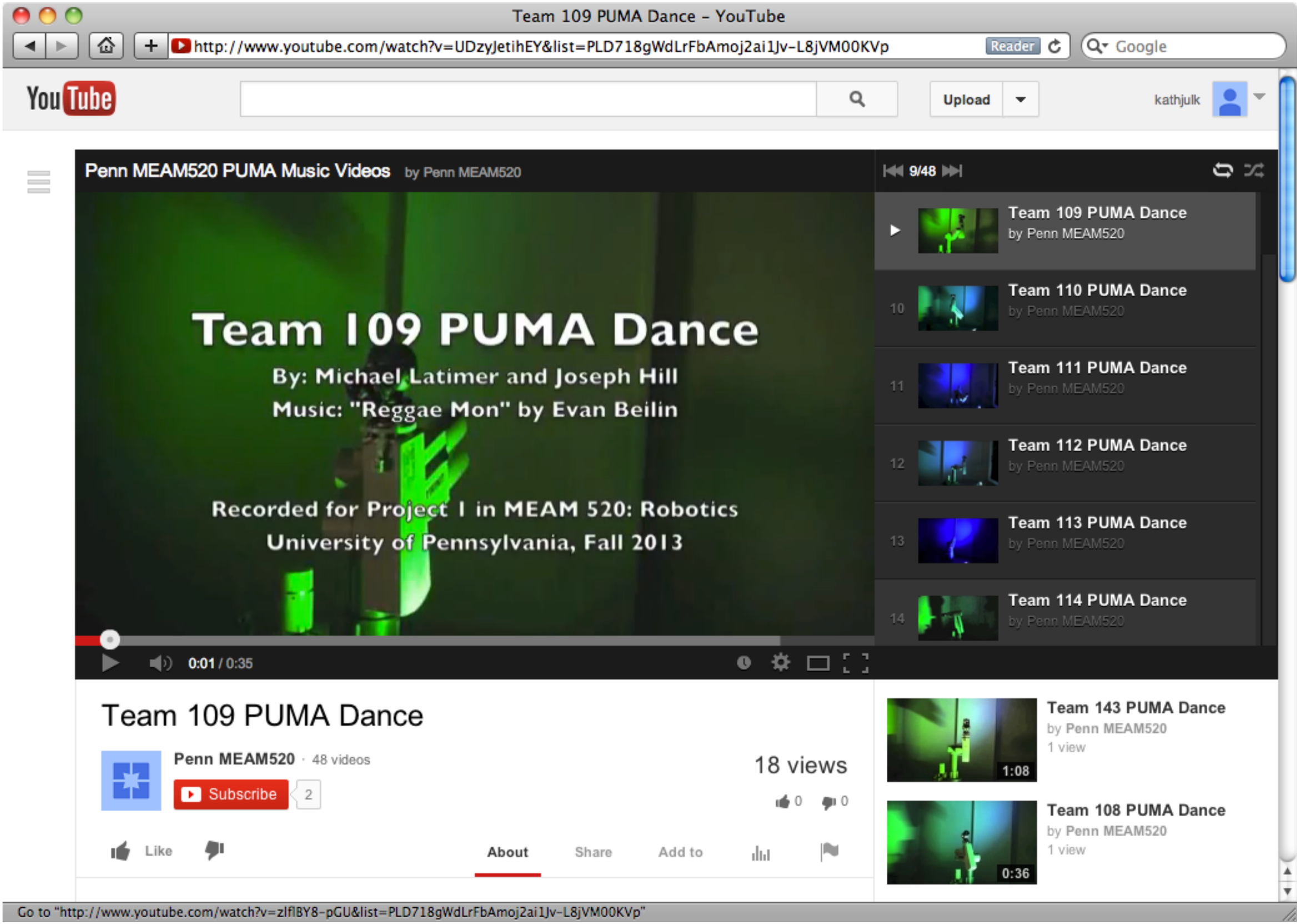
Motor Velocity Plot using Actual Time



This is a nice raw velocity signal. Time to add a low-pass filter!

What questions do you have about this case study?

Let's watch Teams 109–116.



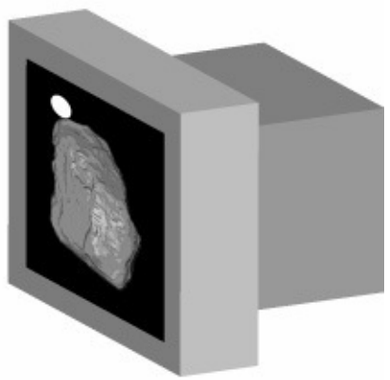
Environment

Real

Assistive Interaction: augments human sensing and/or motion capabilities in real physical environments

Remote

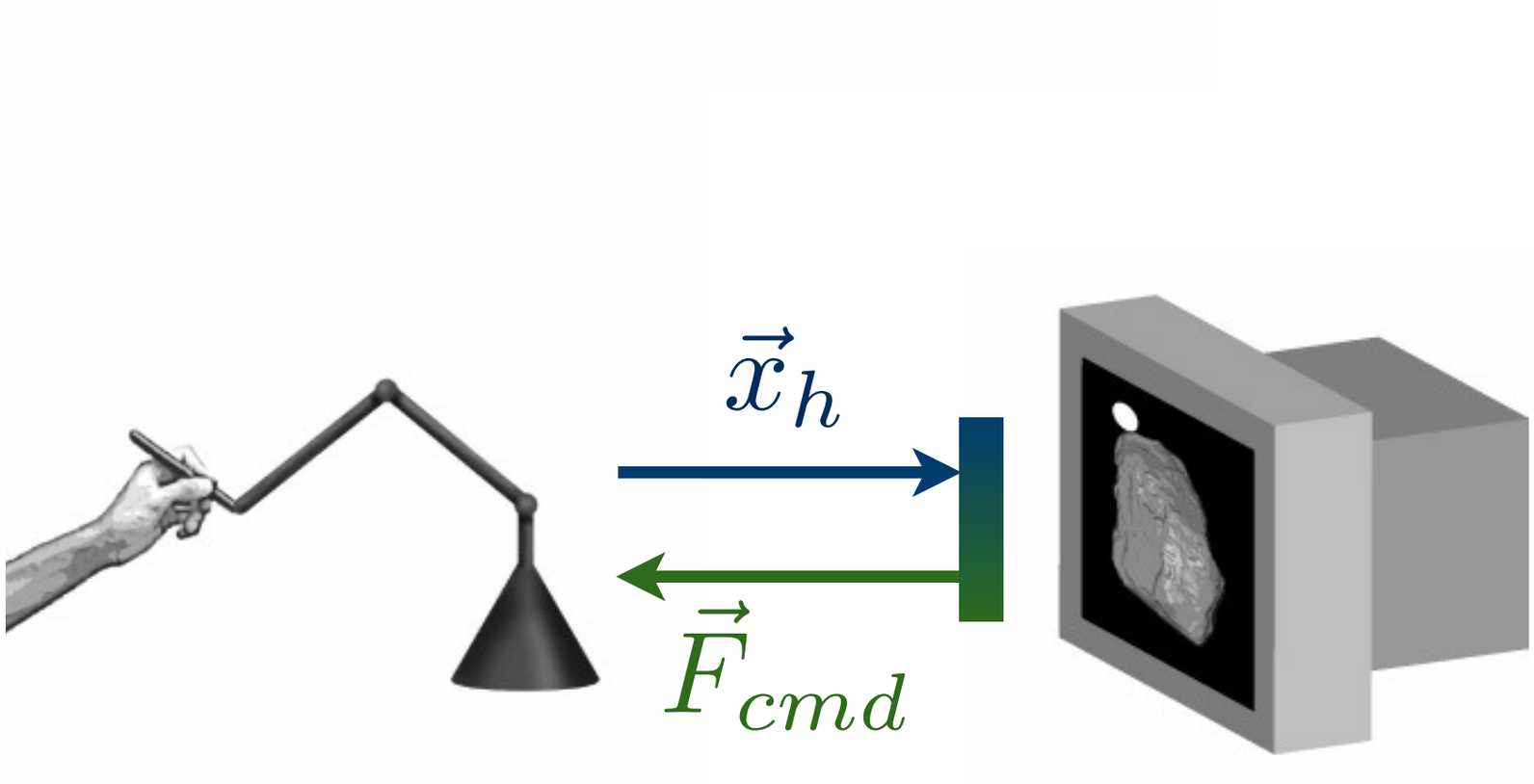
Teleoperation: extends the reach of the human hand to remote, hazardous, unreachable environments



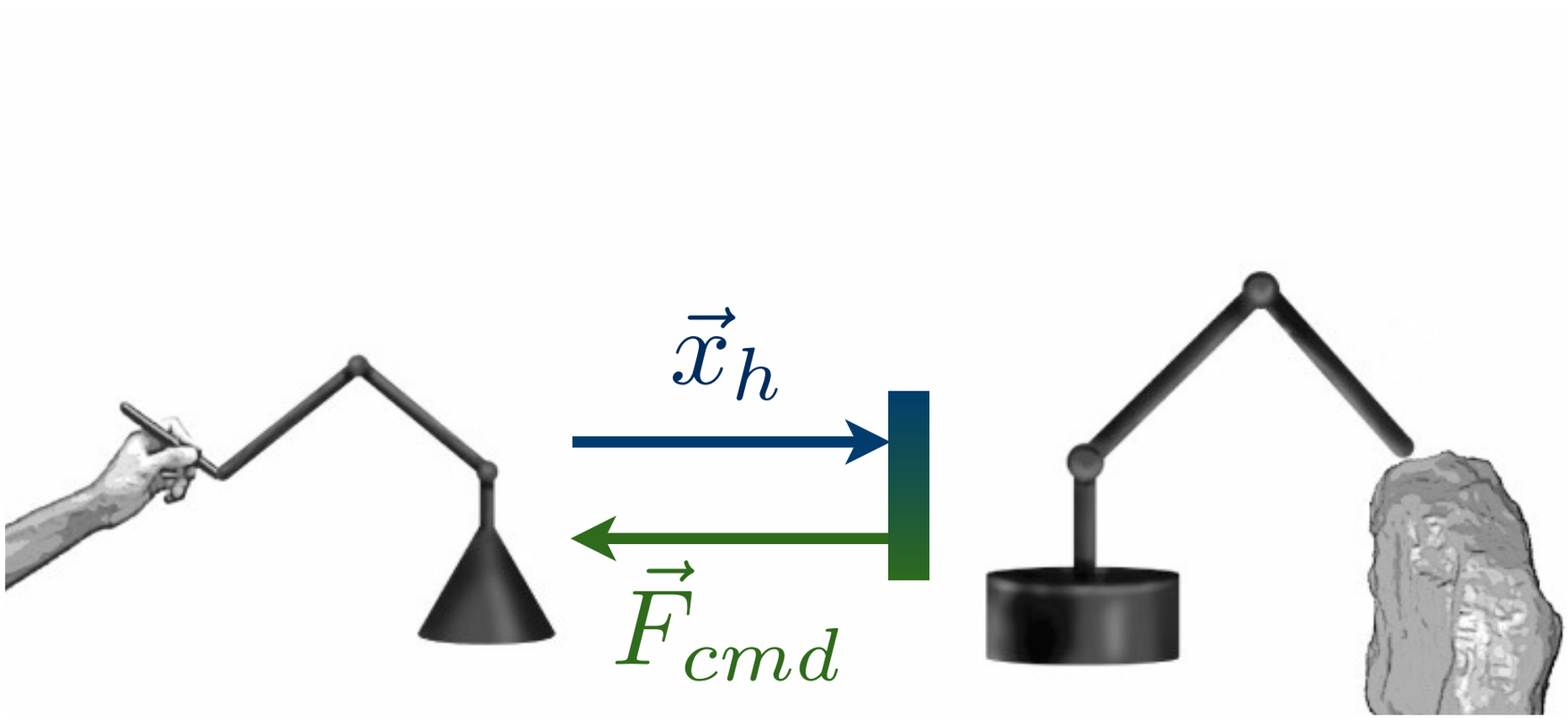
Virtual

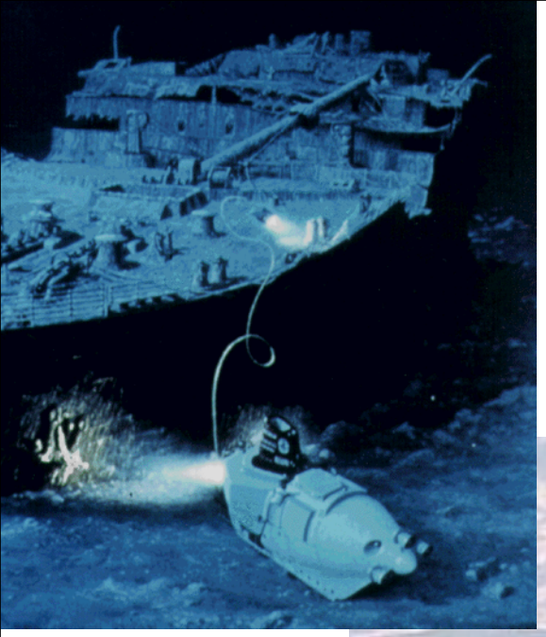
Simulation: enables humans to touch geometric and dynamic computer-based data and models

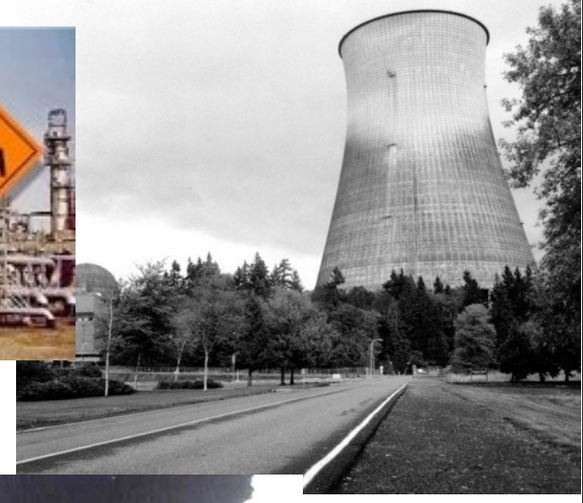
Haptic Virtual Environment



Haptic Remote Environment

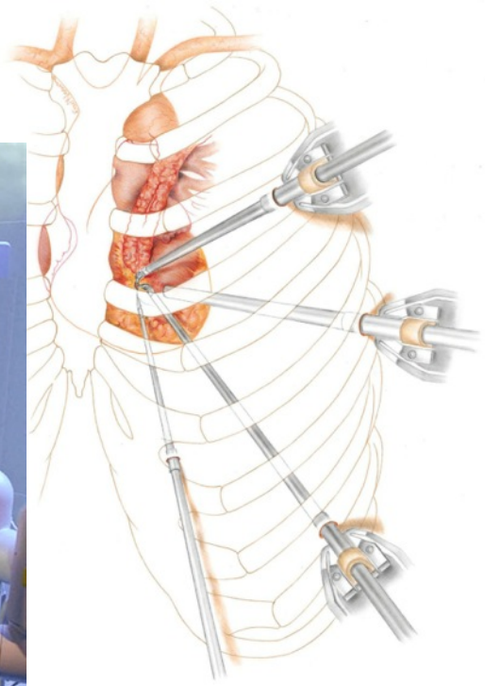




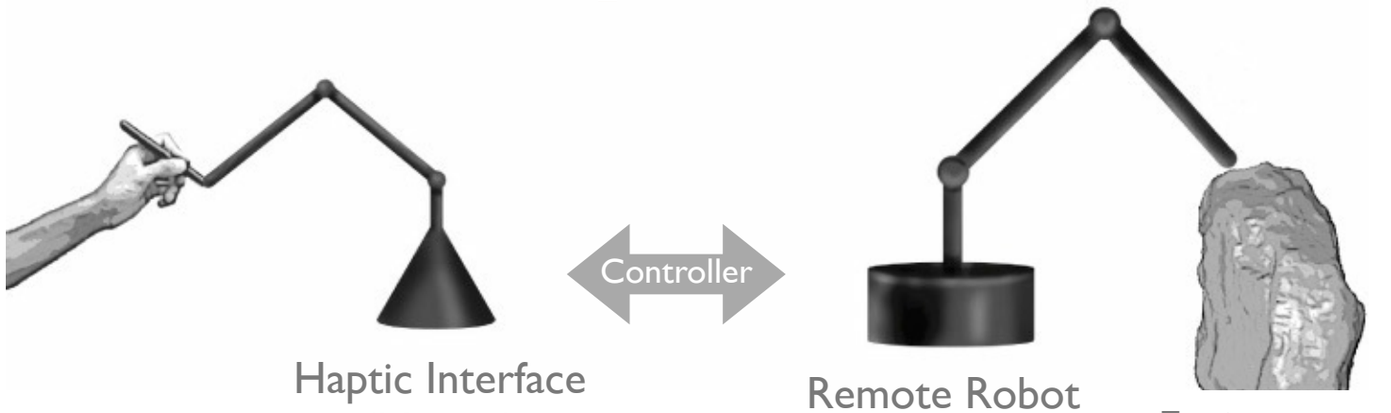


Teleoperation

extends the reach
of the human hand



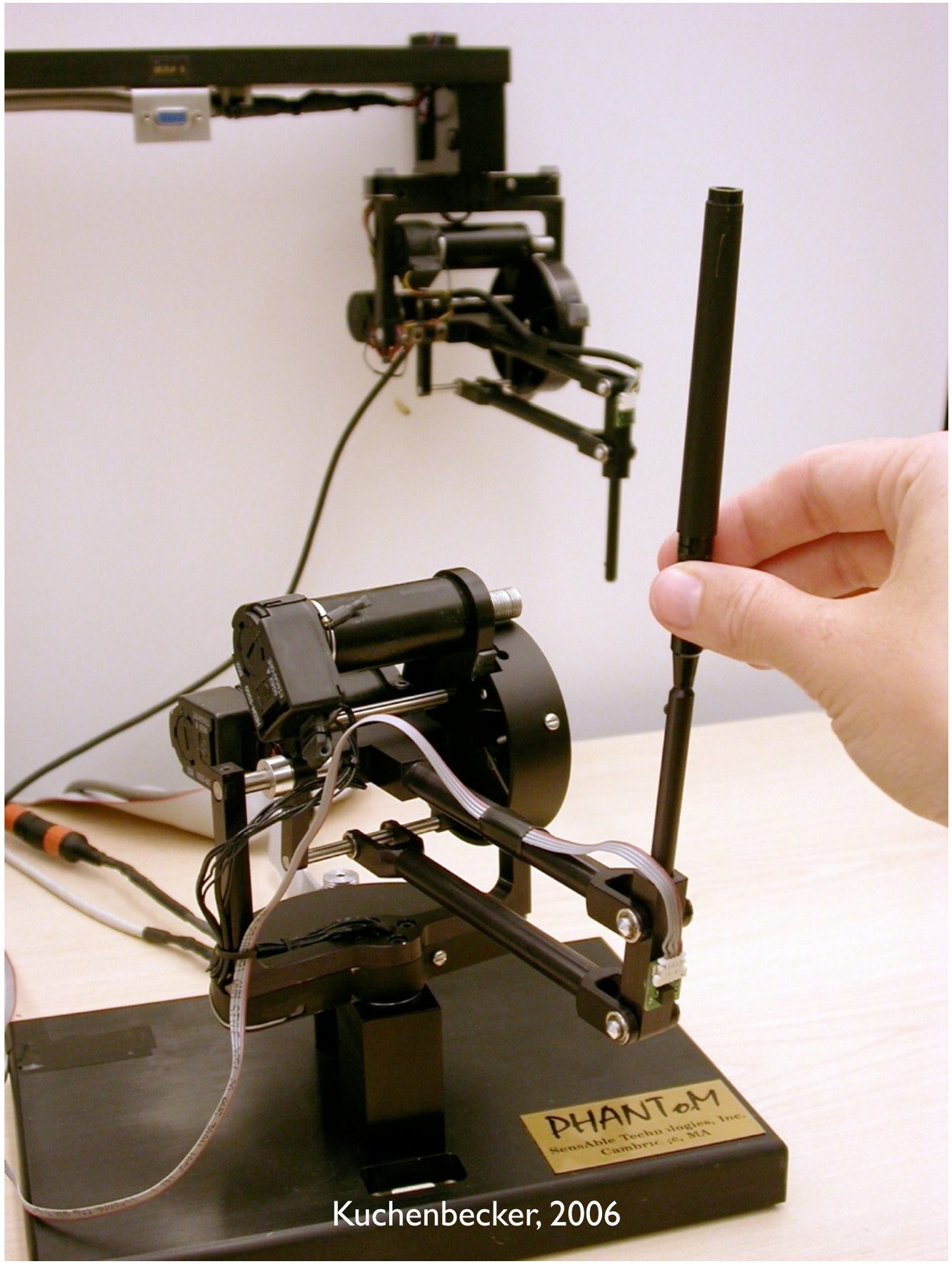
Teleoperation



Mechanical Teleoperation

Goertz, 1952

Modern Teleoperation

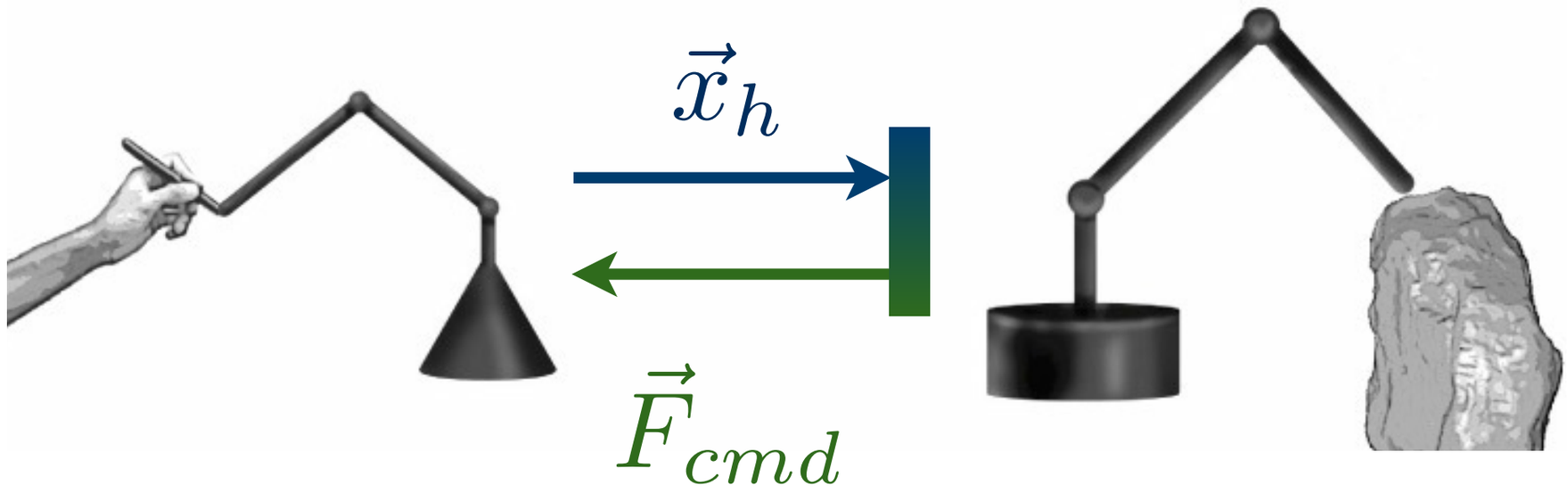


Kuchenbecker, 2006

Robot-Assisted Minimally Invasive Surgery

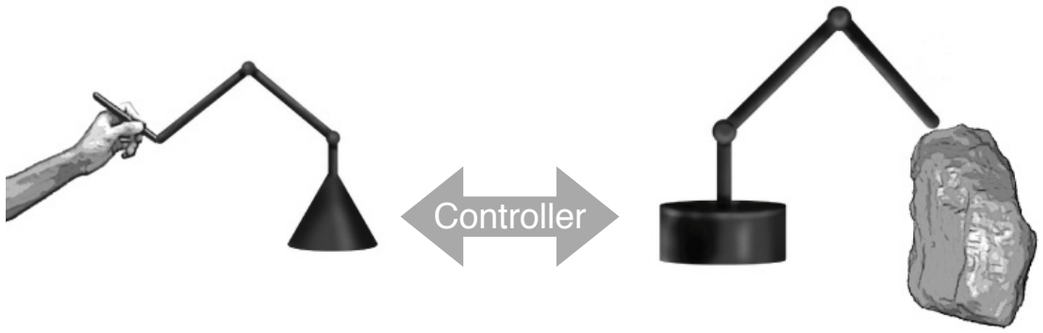
(Intuitive Surgical, Inc., 1998)

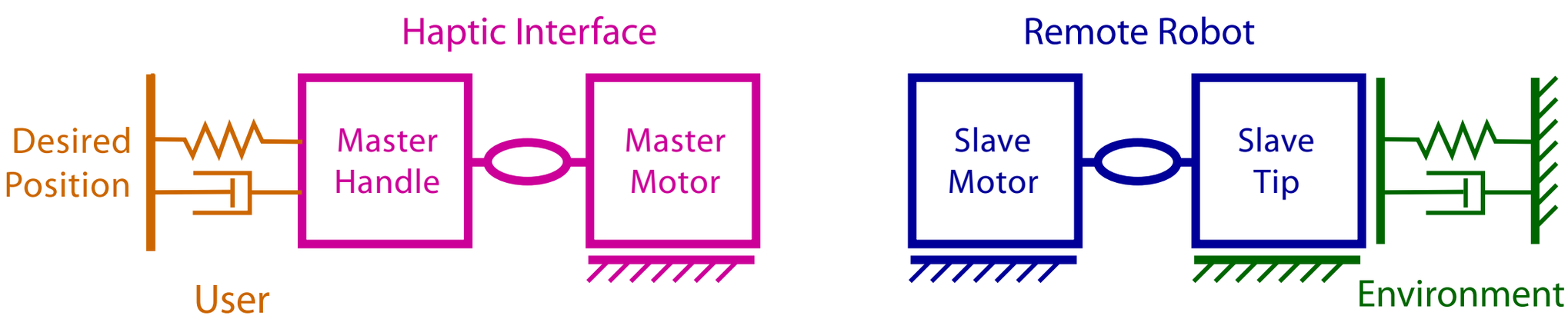
Teleoperation



- Teleoperation has always been tightly intertwined with robotics, especially manipulators.
- Control system design is a primary concern:
  - Stability
  - Transparency

Teleoperation

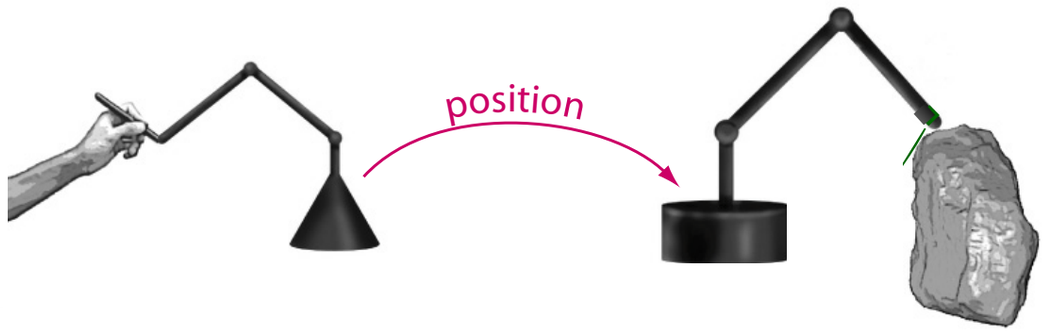




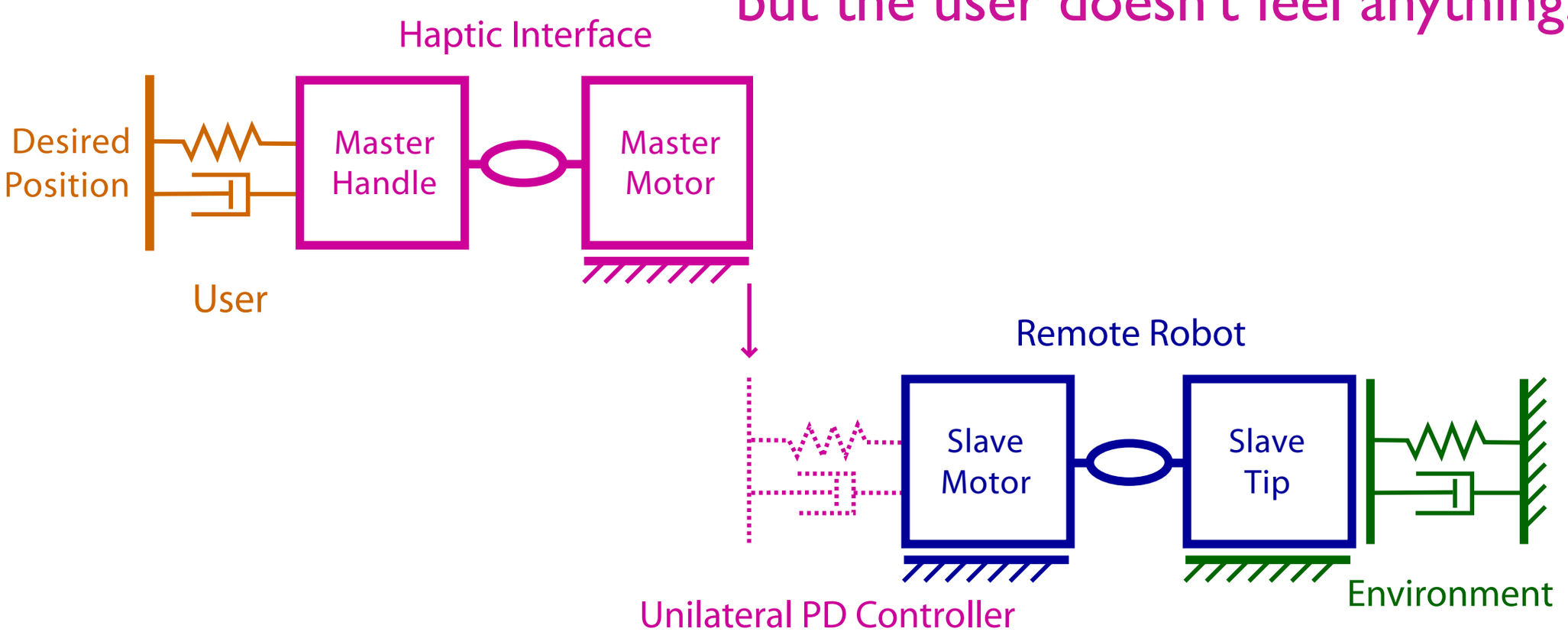
How do we want this system to behave?

As though there was a rigid mechanical connection from the user's hand to the environment.

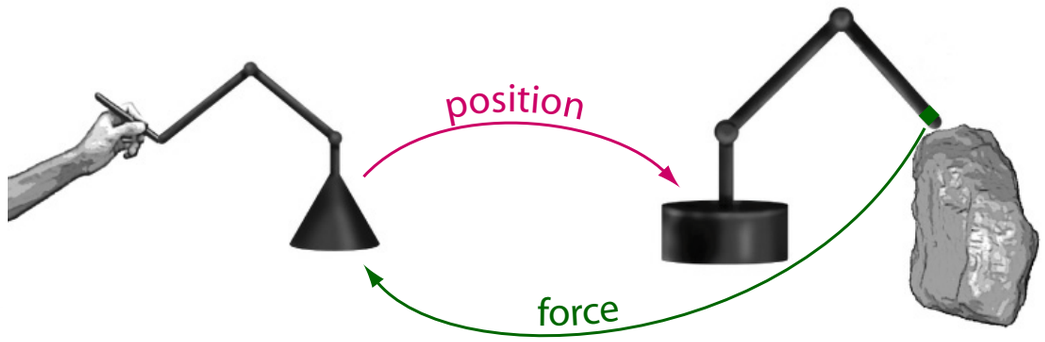
Position-Forward Control



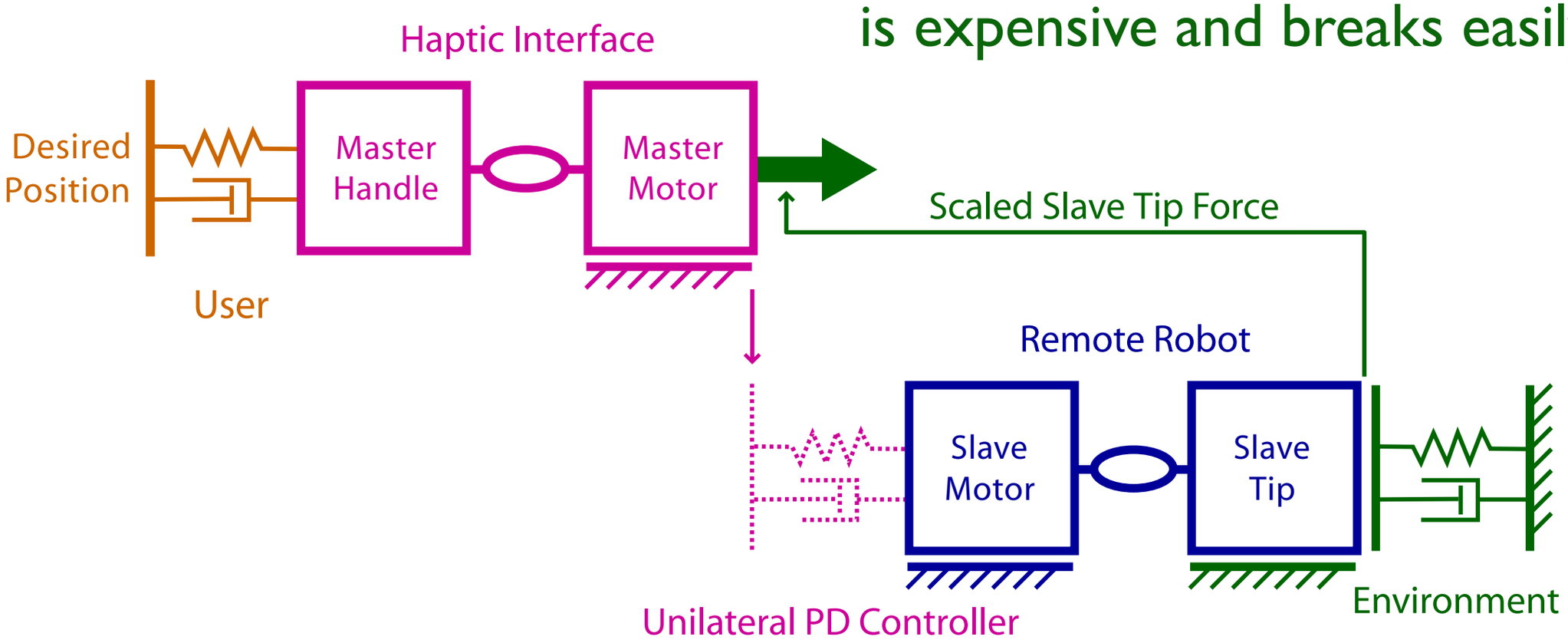
The slave tracks the master's position, but the user doesn't feel anything.



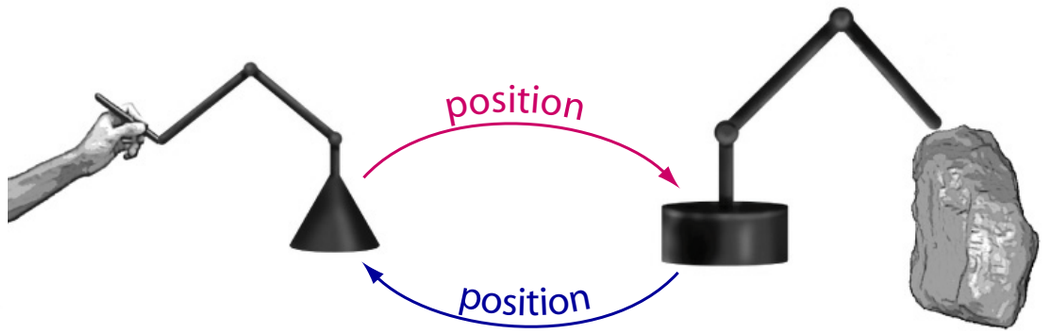
Position-Force Control



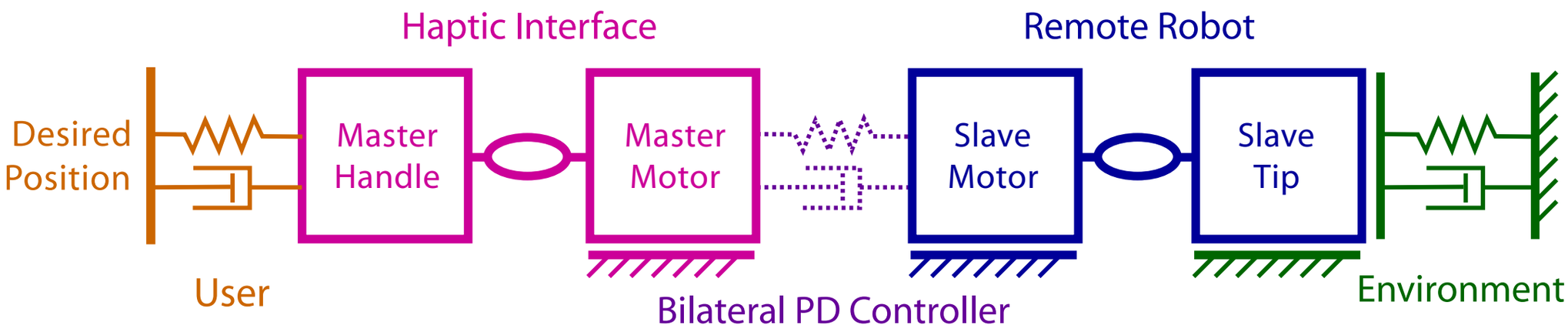
Requires a force sensor, which is expensive and breaks easily.



Position-Position Control



Works pretty well!

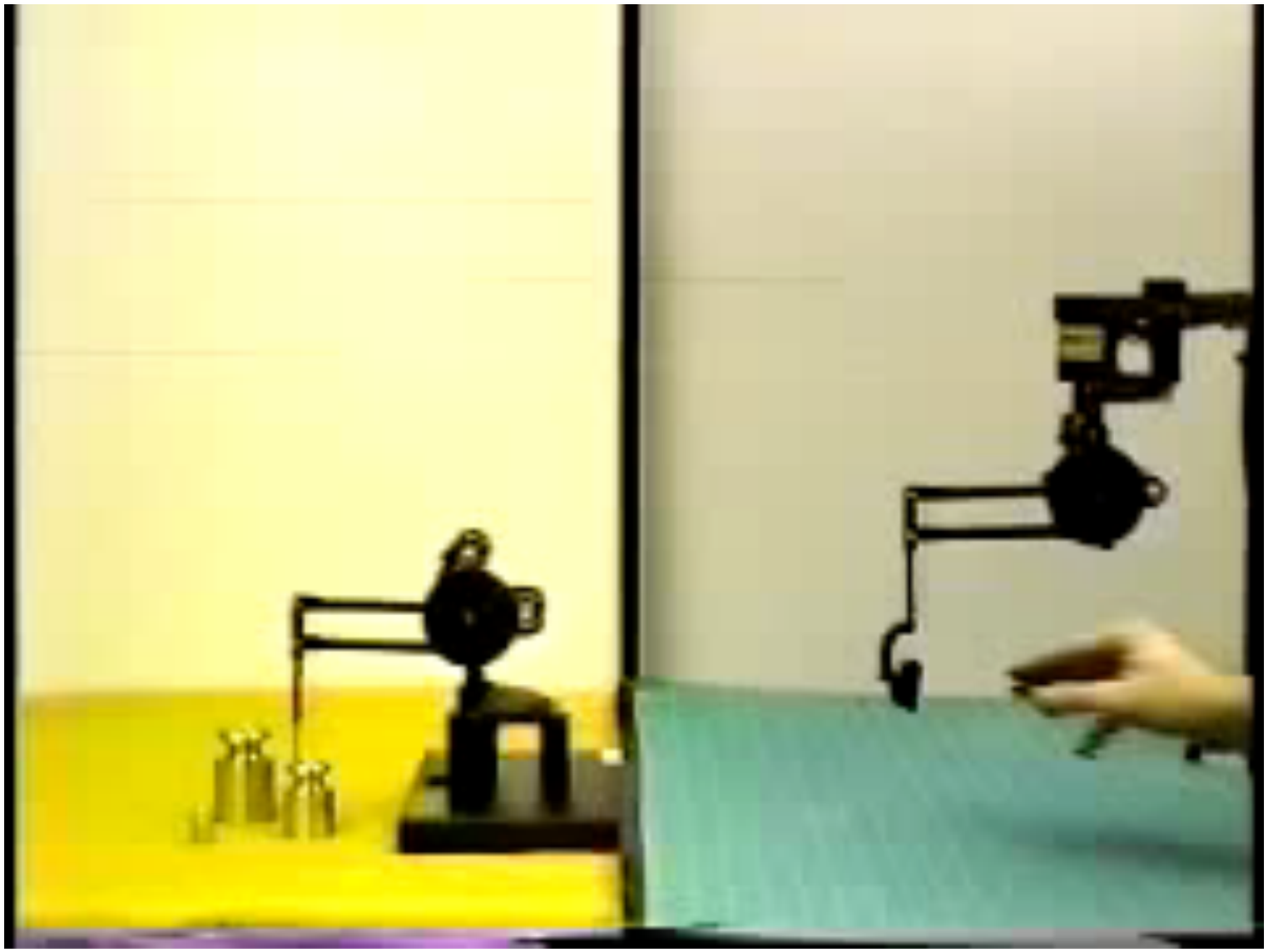


- Each device has a desired state (position and velocity), which is computed from measured states of the other device.
- Separate controllers try to make each device achieve its desired state by using the motors to output forces.

Time Delay

- Time delays between master and slave will make position-position control unstable; time delays are unavoidable when the master and slave are in separate locations.
- When delays are fixed and known, you can use alternative encoding methods, such as wave variables, to avoid instability.
- People then won't notice delays of about 200 milliseconds, but the feel will be deteriorated.
- Can use a predictor to estimate where the user is moving the master, to improve responsiveness.
- Can build a dynamic model of the remote environment and use that to provide haptic feedback locally.
- When time delays become large, operators adopt a "move and wait" approach - give the slave more autonomy.

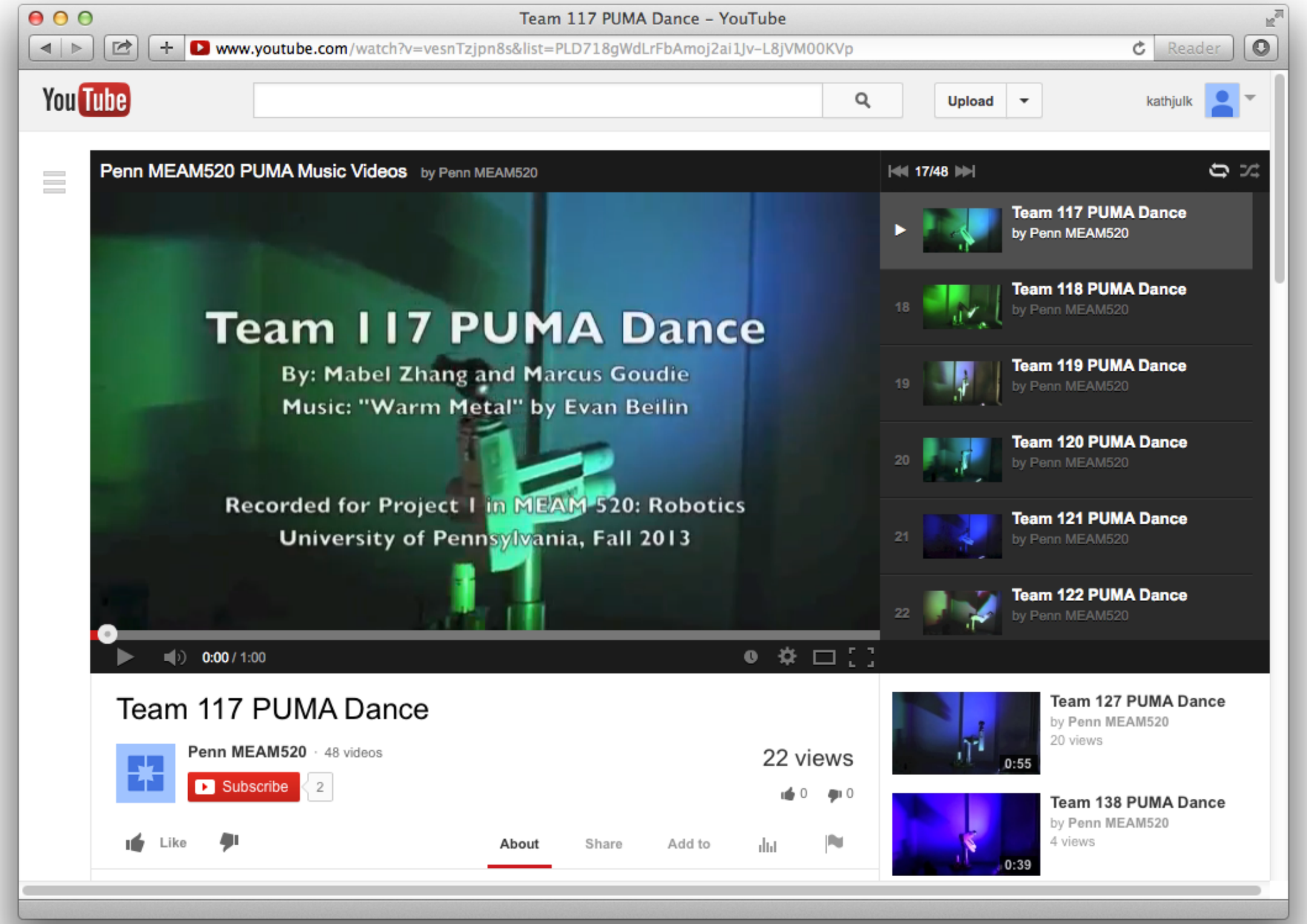
Teleoperation with Time Delay



(Niemeyer and Slotine, 1998)

What questions do you have about teleoperation?

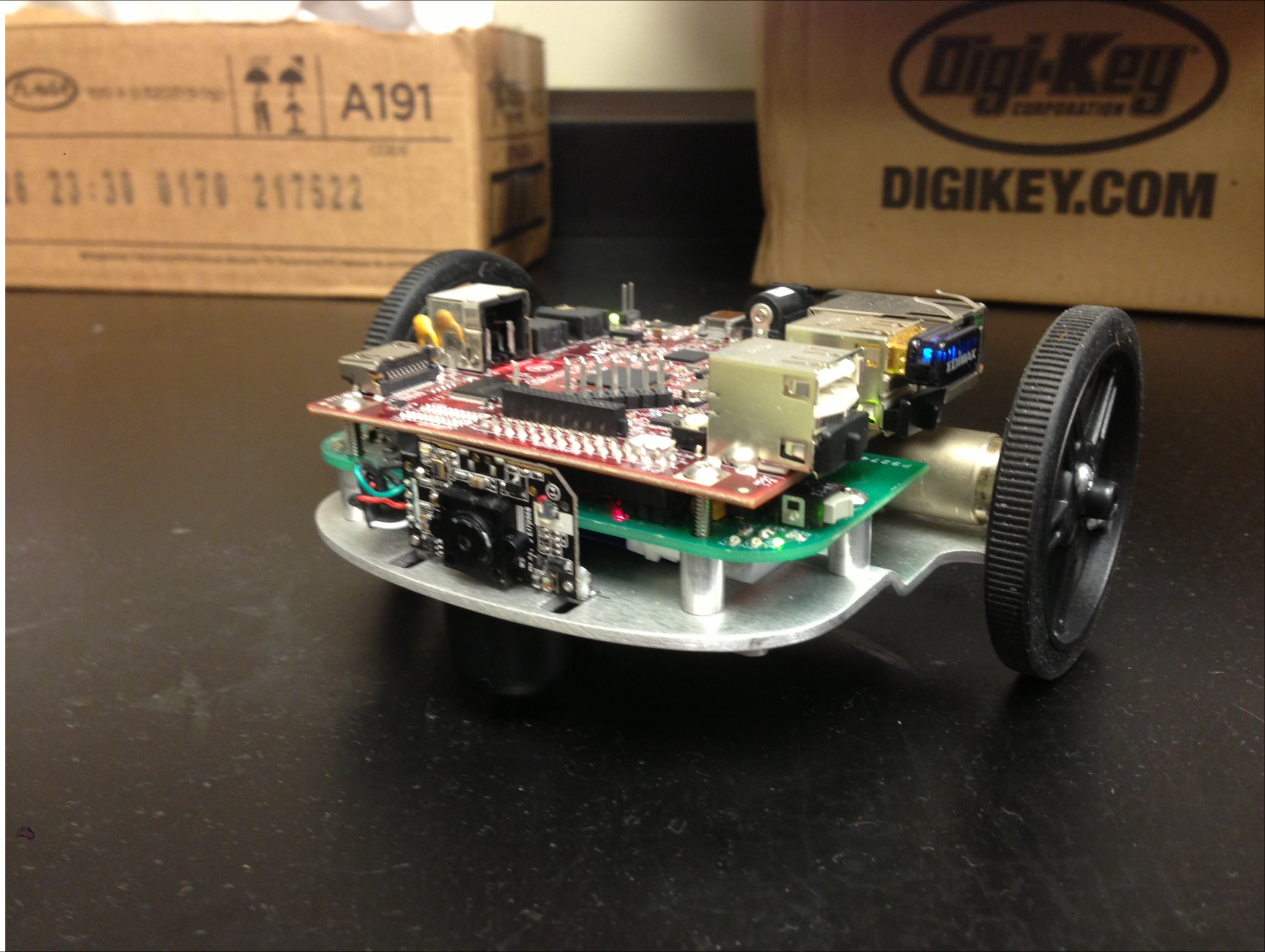
Let's watch Teams 117–123.

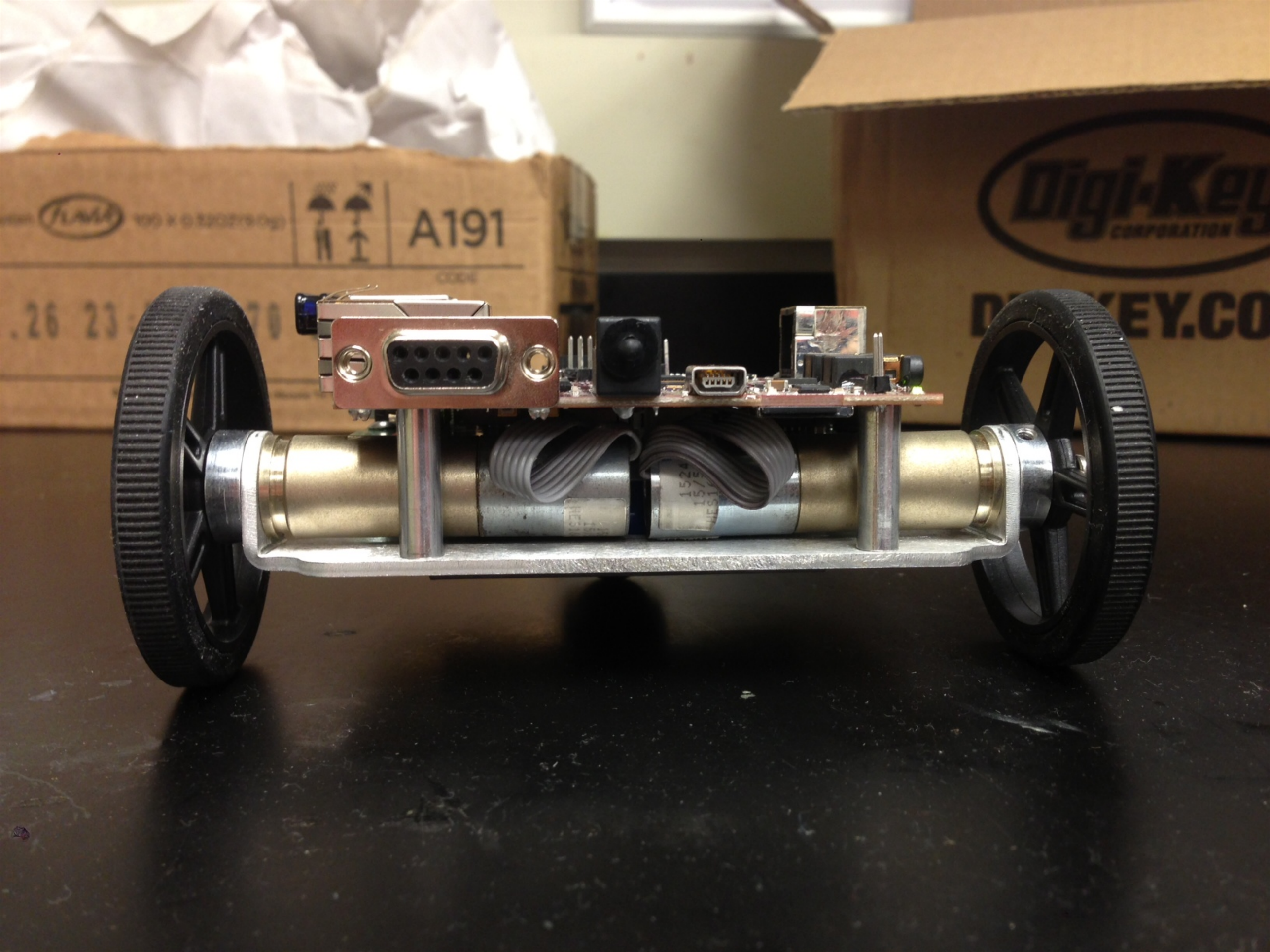


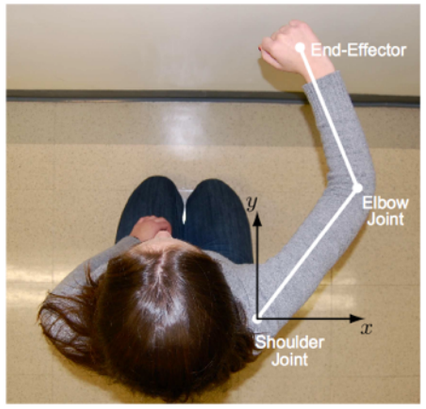
Mobile Robot Demonstration by Two of Our Teaching Staff

Tyler Barkin

Samarth Brahmbhatt


A191
DIGIKEY.COM

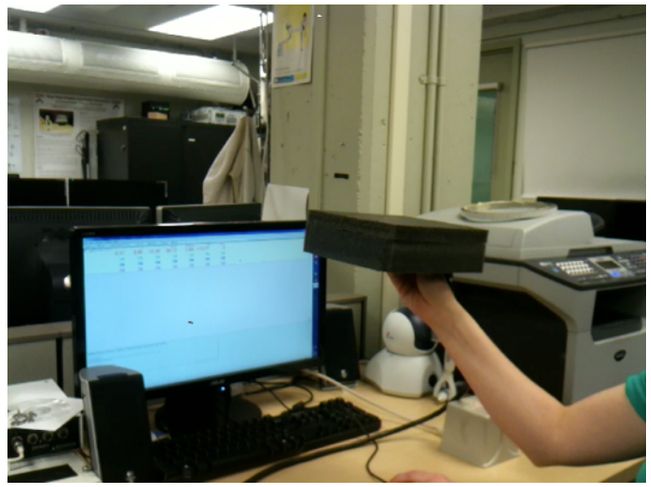

A191

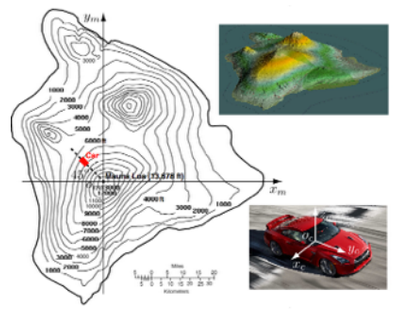


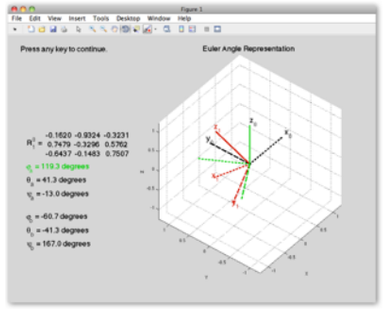
MATLAB

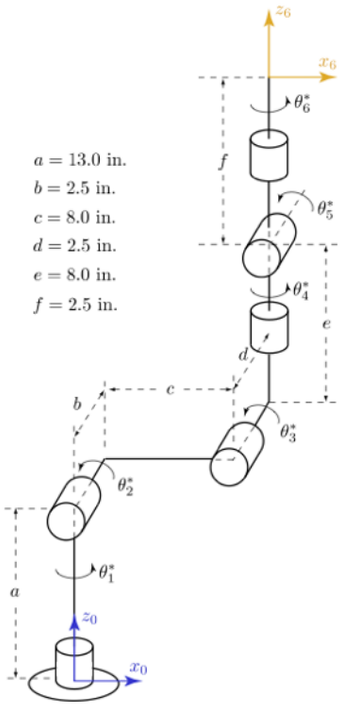
$$\mathbf{H}_2^0 = \mathbf{H}_1^0\,\mathbf{H}_2^1$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ 0\;\;0\;\;0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$





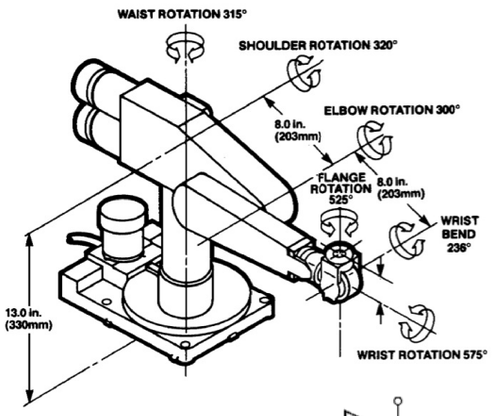




$a = 13.0$ in.
$b = 2.5$ in.
$c = 8.0$ in.
$d = 2.5$ in.
$e = 8.0$ in.
$f = 2.5$ in.

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$\frac{dR}{dt} = S(\vec{\omega})\,R$$



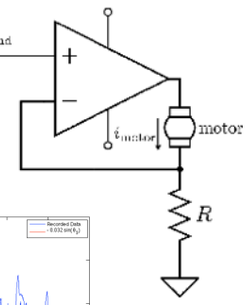
$$-180° < \theta_1^* < 110°$$
$$-75° < \theta_2^* < 240°$$
$$-235° < \theta_3^* < 60°$$
$$-580° < \theta_4^* < 40°$$
$$-120° < \theta_5^* < 110°$$
$$-215° < \theta_6^* < 295°$$

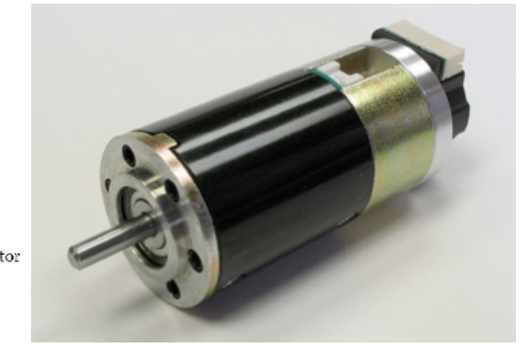
$$A_i = \mathrm{Rot}_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,\mathrm{Rot}_{x,\alpha_i}$$

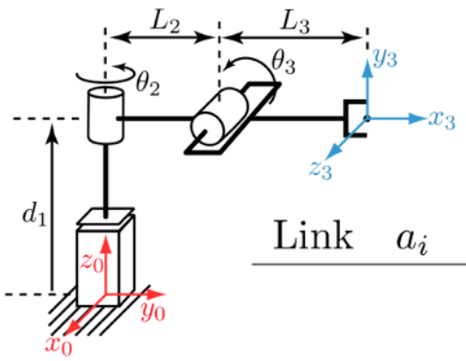
$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

cubic polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$







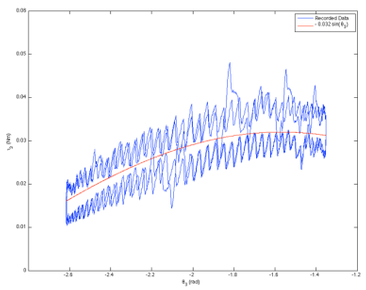
Link	a_i	α_i	d_i	θ_i

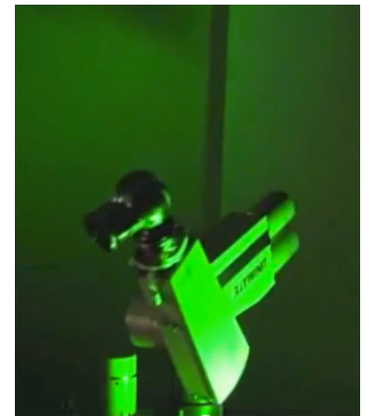
Joint Coordinates — Task Coordinates

Forward Kinematics (FK)

$\theta_1, \theta_2, \theta_3 \rightleftarrows x, y, \alpha$

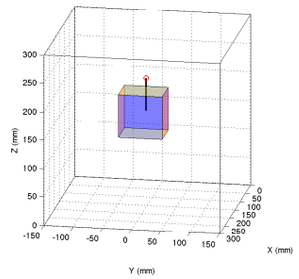
Inverse Kinematics (IK)

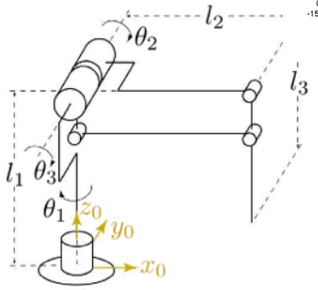




$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

$$\vec{\tau} = J^T(\vec{q})\vec{F} \quad \det(J_v) = 0$$





Thank you for a great semester!