Translation stage design: Procedure

ME 240: Fundamentals of Instrumentation & Measurement • D. H. Kelley and I. Mohammad

Introduction

Engineers often design and build machines that move in controlled ways. One example is a translation stage, a programmable machine that moves a platform or other object to specified locations in space, in one or more dimensions. Dot matrix printers and optical scanners use 1D translation stages to move their print heads or optical sensors back and forth. Most 3D printers use a combination of 2D translation stages to move their print heads and 1D translation stages to raise or lower the printed object as it forms. CNC (computer numerical control) mills and lathes, which are programmable machines for high-precision subtractive (as opposed to additive) manufacturing, likewise use translation stages. In this exercise, you will develop a microcontroller-based algorithm to control a simple 2D translation stage like the one shown in Fig. 1a and use the apparatus to draw the picture shown in Fig. 1b.

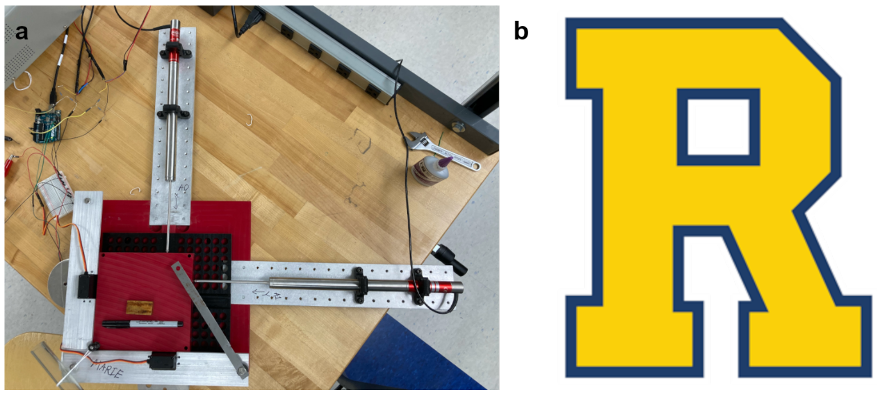

Figure 1: **a,** A simple 2D translation stage. **b,** The picture you will draw by controlling the stage.

To control the stage, you will need to measure its position and send corresponding commands to the motors that move it. You will measure positions with linear variable differential transformers (LVDTs), each of which has a pin that slides in one dimension and produces a voltage that relates linearly to the position of the pin. The model you will use is the RDP Electrosense DCTH2000A, whose specifications can be found here and are linked from the QR code in Fig. 2. You will move the stage with 270° positional servo motors whose position can be controlled with an input voltage. The model you will use is the DS3235SG, whose specifications can be found here and are linked from the QR code in Fig. 2. To read from the LVDTs and write to the servos, you will use an Arduino Uno microcontroller board, whose specifications can be found here and are linked from the QR code in Fig. 2. A microcontroller is programmable and has broad capabilities; we will interface with it via Matlab over USB, making use of its analog inputs to read from LVDTs and its digital ports to read from and write to servo motors. Do not connect any Arduino pin to voltage exceeding 5 V, or else you will destroy the Arduino!

LVDT specifications | servomotor specifications | Arduino specifications | image

Figure 2: QR codes linking to relevant documentation.

Learning goals

- Calibrate LVDTs and use them to measure 2D position.
- Control servomotors electronically, using a microcontroller board.
- Develop an algorithm and write Matlab code to move a 2D translation stage sequentially to a series of positions, drawing a dot at each in order to produce a picture.
- Develop an algorithm and write Matlab code to move a 2D translation stage sequentially to a series of positions, drawing a continuous line in order to produce a picture.

Materials

DC power supply, Arduino microcontroller board, PC with Matlab, translation stage, breadboard wires, Philips screwdriver, calibration block, calipers, paper

Safety

Closed-toed shoes are required.

Determining coordinates for your drawing

For this exercise, your mission is to draw the image in Fig. 1b by placing a dot at each of its vertices. To do it, you will attach a piece of paper to the translation stage, then use Matlab and the Arduino to send the stage sequentially to the location of each vertex.

Start by determining the locations of the dots. The translation stage has a 70 mm throw (distance from minimum location to maximum location) in each direction, and the image in Fig. 1b is sized to fit, so one way to determine the vertex coordinates is simply to measure Fig. 1b with a ruler. You may be able to measure more precisely by importing the image file (also linked from a QR code in Fig. 2) into Matlab and using the Data Cursor tool to determine the coordinates (in pixels) of each vertex. However you measure, you may need to scale and/or offset your measurements to make them fit in the 70 mm $\times$ 70 mm region. In your report, include a table listing your vertex coordinates (using real units, e.g. mm or inches, not pixels) and explain how you got them.

Controlling a servo motor via Arduino

All the components should already be connected. Start by verifying the wiring.

- The Arduino microcontroller board should be connected to a PC.
- The x-direction servomotor has an white signal wire that should be connected to Arduino digital pin `D3`. The white signal wire of the y-direction servomotor should go to pin `D11`. (Be sure to use pins capable of pulse width modulation.)
- Each servomotor power wire, which is red, should connect to +6 V via a fuse and should *not* connect to the +5 V pin on the Arduino.
- Each servomotor ground wire, which is black, should connect to ground on the Arduino board.

The control software uses a range from 0 to 1, where 0 corresponds to the minimum servo rotation angle and 1 corresponds to the maximum servo rotation angle. Verify that you are able to send the servomotor to a position between 0° and 270° and are able to read its current value. Read and write to the x-direction servo motor using Matlab commands like

```
a = arduino(); % initialize Arduino
servoX = servo(a,'D3') % initialize servo on Arduino pin D3
readPosition(servoX)
writePosition(servoX,1) % sends the servomotor to its 270° position
readPosition(servoX)
writePosition(servoX,0) % sends the servomotor to its 0° position
readPosition(servoX)
writePosition(servoX,0.52) % stop
```

Test the y-direction servo motor using similar commands.

Calibration: Relating LVDT voltage to position

An LVDT produces a voltage that relates linearly to the position of its pin. You will need to quantify that relationship, for each of the LVDTs in the translation stage, in order to draw a picture. First, verify the wiring of the LVDTs connected to the Arduino microcontroller board.

- The x-direction LVDT has a white signal wire that should be connected to the Arduino analog input port `A0`. The signal wire of the y-direction LVDT should go to port `A5`.
- LVDT ground wires are black and should be connected to the Arduino ground.

Note that we are using the LVDT output that produces a 0 to 10 V signal (see the specifications), but since the range of an Arduino analog input is 0 to 5 V, we have added a 50% voltage divider.

Define the position where the LVDT pins are maximally extended and the stage is in contact with the motor mount as the origin. Read the voltage when the x-direction pin is maximally extended using Matlab commands like

```
a = arduino(); % initialize Arduino
readVoltage(a,'A0') % read voltage on A0
```

Then, slide the pin out of the way, insert the calibration block between the pin and the stage, and release the pin so its tip contacts the block. Record the voltage. Repeat twice more, rotating the block so that you make one measurement for each of its three different lateral dimensions (length, width, and height). Use calipers to measure those dimensions. Plot the four data points, with voltage on the horizontal axis and displacement on the vertical axis. Fit a line to the points, add the fit line to the plot, and note the slope and intercept. You now have a calibration: the formula for this line can be used to translate voltages into positions or vice versa. Repeat these steps for the other LVDT, producing a corresponding plot, to calibrate it as well. Include these figures in your report.

Single-goal, one-dimensional translation stage control

Now, write a Matlab script to send the translation stage to a single "goal" position, in one dimension. One possible algorithm: read a voltage from the x-direction LVDT, use that voltage and your calibration to calculate the x component of the stage's position, compare that position to the goal (e.g., the x coordinate of one of the vertices you measured), and spin the x servomotor a small amount to advance toward the goal. Repeat these steps in a loop until the goal is reached (within some tolerance, of course). Whether the servomotor angle is increased or decreased should be decided with each iteration of the loop, depending on whether the current stage position is greater or less than the goal. When you use `writePosition(servoX, value)`, be sure that `0` $\leq$ `value` $\leq$ `1`, or else an error will result.

When reading from the LVDT using an analog input port on the Arduino, rise time must be considered. Together, the LVDT and analog input circuitry constitute a measurement system with a finite rise time. The Arduino, like most analog-to-digital conversion devices, has multiple input channels but only a single analog-to-digital converter circuit. Channels are measured one after the other, by switching the circuit from port to port. (Converter circuits are expensive.) Every time the Arduino switches to a new analog input channel, some time is required for the voltage to stabilize. That time is relatively long when an LVDT is connected, because an LVDT is comprised of coils of wire that have large inductance. Your script will not produce accurate measurements unless, after switching channels, it discards the first measurement and pauses briefly before making another measurement. You can use commands like

```
readVoltage(a,LVDT_x_pin); % initialize analog channel
pause(0.001); % wait 1 ms
Vx = readVoltage(a,LVDT_x_pin); % now keep the reading
```

Try your script with two or three different "goal" positions to make sure it works. Submit the script with your lab report.

One-dimensional translation stage control

Now, write a new script, based on the previous one, that does the same thing repeatedly, for multiple “goal” positions. To do it, wrap another loop around the outside of the previous algorithm, with each loop iteration pursuing a new goal. Make sure your script works with a list of at least three goals. Change the goal values, perhaps requiring the servomotor to reverse directions, and run the script again to be sure your code is correct. Submit the script with your lab report.

Two-dimensional translation stage control

Now, write a new script, based on the previous one, that moves the stage to a series of two-dimensional (x, y) “goal” positions. Probably the simplest algorithm involves moving just one servomotor at a time, for example by first matching the x component of the goal position, then matching the y component. Test your script with a list of at least three goals. Once you are confident that the script is correct, provide all the vertex coordinates as goals, attach a piece of paper to the translation stage, and use your script to draw the vertices of image in Fig. 1b. Connect the vertices by hand afterward if you like. Submit the script and drawing with your report.

Continuous drawing

Finally, write a script to draw the same picture *continuously*, keeping the pen always in contact with the paper and moving the translation stage directly from vertex to vertex. Diagonal lines will be the tricky part! Submit the script and drawing with your report.