Laboratory 6:

Cal Climber Part IV

Model-Based Hill Climb in Statecharts

Authors:
Jeff C. Jensen (National Instruments)
Trung N. Tran (National Instruments)

Instructors:
Edward A. Lee
Sanjit A. Seshia

University of California, Berkeley
EECS 149

February 28, 2011

1 Introduction

1.1 Goals

In this lab, you will:

1. Perform sensor calibration on the Cal Climber.
2. Using a state machine, program the Cal Climber to autonomously climb a hill.

This lab should take at least 3 hours to complete. This is the last of a four-week sequence. There is no required reading for this lab.

1.2 Equipment

1. PC computer running Windows XP.
2. National Instruments LabVIEW 2010
   (a) LabVIEW 2010 Full Development System
   (b) LabVIEW Statecharts Module 2010
   (c) LabVIEW Robotics 2010
   (d) LabVIEW Robotics Simulator 2010 (beta)
   (e) LabVIEW Real-Time 2010
3. Cal Climber
   (a) iRobot Create
   (b) National Instruments single-board RIO (sbRIO) NI-9632
   (c) Asus WL-330ge wireless router
   (d) Analog Devices ADXL-322
   (e) DC voltage regulator

1.3 Sensor Calibration

You have had experience calibrating two accelerometers, namely the WiiMote in LabVIEW, and the ADXL-322 in C. In this lab, we provide you with a calibration procedure that allows for the accelerometer on the Cal Climber to be placed in *any* orientation. The procedure follows the WiiMote lab.

Let:

- $\theta : \mathbb{R} \longrightarrow (-\pi, \pi]$ be the orientation of the robot, with $\theta(t) = \frac{\pi}{4}$ if the robot is oriented uphill, and positive rotation in the clockwise direction.
- $\vec{a}_{\mathrm{g}} = \left(a_{\mathrm{g_x}}, a_{\mathrm{g_y}}\right) : \mathbb{R} \longrightarrow [-g, g]^2$ be the component of gravity that is in the plane of the robot, according to our coordinate system.
- $\vec{a}_{\mathrm{meas}} = (x, y) : \mathbb{R} \longrightarrow \mathbb{Z}^2$ be the quantized measurement of the accelerometer, which will in general represent a coordinate system that is different from our ideal coordinate system. We assume the accelerometer measures only gravity.
- $x_0, y_0 \in \mathbb{Z}$ be the accelerometer measurements when the robot is on level ground, so that $\vec{a}_{\mathrm{meas}}(t) = (x_0, y_0) \Leftrightarrow \vec{a}_{\mathrm{g}}(t) = \vec{0}$. x_0 and y_0 are referred to as *0g offsets*.
- $x_1, y_1 \in \mathbb{Z}$ be the change accelerometer measurements of each axis from 0g to 1g, so that $\forall t \left\{|x(t) - x_0| \leq x_1 \wedge |y(t) - y_0| \leq y_1\right\}$. x_1 and y_1 are referred to as *+1g offsets*.
- $x_{\mathrm{hill}}, y_{\mathrm{hill}} \in \mathbb{Z}$ be any accelerometer measurement when the robot is oriented uphill, so that $\angle \vec{a}_{\mathrm{meas}}(t) = \angle\left(x_{\mathrm{hill}} - x_0, y_{\mathrm{hill}} - y_0\right) \Leftrightarrow \angle \vec{a}_{\mathrm{g}}(t) = \frac{\pi}{4}$.
- $\theta_{\mathrm{c}} \in (-\pi, \pi]$ be the rotation between the ideal and actual accelerometer measurements, so that $\forall t \left\{\angle \vec{a}_{\mathrm{g}}(t) - \angle\left(\vec{a}_{\mathrm{meas}}(t) - (x_0, y_0)\right) = \theta_{\mathrm{c}}\right\}$. θ_{c} is referred to as the *correction angle*.

Then

$$\theta_{\mathrm{c}} = \angle\left(x_{\mathrm{hill}} - x_0, y_{\mathrm{hill}} - y_0\right) = \tan^{-1}\left(\frac{y_{\mathrm{hill}} - y_0}{x_{\mathrm{hill}} - x_0}\right), \tag{1}$$

and

$$\vec{a}_{\mathrm{meas}}(t) = \begin{pmatrix} x_1 & 0 \\ 0 & y_1 \end{pmatrix} \left[\left[\begin{pmatrix} \cos(\theta_{\mathrm{c}}) & -\sin(\theta_{\mathrm{c}}) \\ \sin(\theta_{\mathrm{c}}) & \cos(\theta_{\mathrm{c}}) \end{pmatrix} \vec{a}_{\mathrm{g}}(t)\right]\right] + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \tag{2}$$

where $[[\ \cdot\ ]]$ denotes the nearest integer function. We can approximate the acceleration in our ideal coordinate system by reconstructing $\vec{a}_{\mathrm{g}}$,

$$\vec{a}_{\mathrm{g}}(t) \approx \begin{pmatrix} \cos(\theta_{\mathrm{c}}) & \sin(\theta_{\mathrm{c}}) \\ -\sin(\theta_{\mathrm{c}}) & \cos(\theta_{\mathrm{c}}) \end{pmatrix} \begin{pmatrix} \frac{1}{x_1} & 0 \\ 0 & \frac{1}{y_1} \end{pmatrix} \left[\vec{a}_{\mathrm{meas}}(t) - \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}\right], \tag{3}$$

where approximation is used to reflect quantization error. Equation (3) shows that calibration points may be used to reconstruct the acceleration vector in our ideal coordinate system, regardless of the actual orientation of the accelerometer. We have implemented this transformation in your template project for this lab.

1.4 Sensor Conditioning

After calibrating your accelerometer, it is a good idea to use a lowpass filter to prevent chattering and to isolate gravity from coordinate acceleration. We suggest using an *exponential moving average*, $h : [\mathbb{Z} \longrightarrow \mathbb{R}] \longrightarrow [\mathbb{Z} \longrightarrow \mathbb{R}]$, where for $\alpha \in [0, 1]$

$$(h \circ x)[n] = (1 - \alpha)\left((h \circ x)[n - 1]\right) + \alpha x[n]. \tag{4}$$

Smaller α place more weight on previous values of $x[n]$, and larger values place more weight on more recent values. We have implemented this filter in your template project for this lab.

2 Procedure

2.1 Load the Template Project

We have made a number of improvements to the template project you used in the previous lab:

- Corrected scale and frame of reference for the simulated accelerometer.
- Fixed the "Wheel Speed" control on the host VI, which was previously uneditable.
- Added an instruction to stop the iRobot when the target VI terminates.
- Increased the update rate between the target and the host (from 4 Hz to 10 Hz).
- Fixed debouncing of the "Play" button in the Statechart diagram. The update rate of the iRobot sensors may be too slow to have noticed, but it was not correctly debounced in the previous version.
- All angles are wrapped to between $[-180°, 180°]$ where appropriate.
- Added a "Stop Target" button to the target VI.
- Added a new simulation environment.
- Added accelerometer calibration and conditioning VIs.
- Fixed the "Wall" boolean sensor in the simulated iRobot, which had been inverted.
- Increased the boolean wall and cliff sensor threshold to 50mm.

Download the file **lab06files.zip** from the course website, and extract it to the **U:\EECS149** folder. The files for this lab *must*, after extraction, reside in the **U:\EECS149\Lab6** folder. Replace the template Statechart diagram with the diagram you authored in Lab 5. Copy the file **U:\EECS149\Lab5\Cal Climber\Cal Climber State Machine.lvsc** to **U:\EECS149\Lab6\Cal Climber\Cal Climber State Machine.lvsc**.

Open the project file **U:\EECS149\Lab6\Cal Climber\Cal Climber.lvproj** to load your project.

2.2 Simulator

Run the simulator to verify your controller from the previous lab works as expected. You will notice a new simulated environment.

We did not include noise or coordinate acceleration in modeling the accelerometer, nor is its measurement passed through an exponential moving average filter. If you use a threshold to identify when the robot is on an incline, you may need to change the threshold when deploying on your target to match the actual incline of the ramp.

Augment your state machine to differentiate between level ground and an incline. On level ground, your robot should perform as it did in Part III of this lab. On an incline, it should climb to the top, while avoiding cliffs and obstacles around the way.

Note: In simulation, the accelerometer only measures gravity, and not coordinate acceleration. The ramp is at a 10% incline.

Hint: Add a threshold to the StateData control that can be used to determine whether or not the robot is on an incline. Calculate the magnitude of the accelerometer to compare with this value. Any time you change the values of the StateData control, you must right click on the control and select "Data Operations, Make Current Value Default" for the value to be used at runtime.

2.3 Deploy Your Controller

2.3.1 Configure Equipment

Verify that you can ping the static IP address of your target when it is powered on and has had time to join the wireless network.

2.3.2 Calibrate the Accelerometer

Under the target in the Project Explorer, open "Accelerometer Calibrate.vi". This VI runs on the target and is used to record calibration points you will integrate into the Cal Climber target VI. Write the values down when you have completed calibration and verify the correct output in the accelerometer graphs.

2.3.3 Review and Modify the Top-Level Target VI

Review the block diagram of **Cal Climber.vi**. Don't be intimidated, there is a fair amount of code, but it is fairly straightforward. You will see four concurrent timed loops, each configured to run at a different rate. The first loop polls the accelerometer. The second loop polls the iRobot sensors. The third loop executes your Statechart model and sends drive commands to the robot. The fourth loop is used to communicate debugging information back to the host.

You only need to make one modification to this VI, in the accelerometer polling loop. Change the inputs to the accelerometer conditioning VI to match the calibration points you gathered in the previous step. When correctly scaled and filtered, you should see acceleration graphs on the host VI that matched what you saw in the calibration step.

If you do not want to use the filter and calibration we outline in this guide, and instead incorporate your calibration from a previous lab, you may do so in this loop. Keep in mind that, in order to match the behavior in the simulator, you must translate and scale the raw accelerometer values to fall within $[-1\text{g}, 1\text{g}]$.

2.3.4 Test and (if necessary) Modify Your Controller

Run your robot through the obstacle course and ramp to verify your controller is behaving correctly. If needed, revise your Statechart diagram. Take note of any changes from the controller you used in simulation.

3 Acknowledgment

We thank Godpheray Pan, Ansley Li, and Travis Mansfield of National Instruments for their thorough and timely modeling of the Cal Climber in the LabVIEW Robotics Simulator. We also wish to thank the University of California, Berkeley, Department of EECS Instructional Support Group (ISG) and Electronics Support Group (ESG) for their continued efforts in supporting this lab.

References

[1] E.A. Lee and S.A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, digital version 1.03. Berkeley, California, 2010. Available: http://leeseshia.org/releases/LeeSeshia_DigitalV1_03.pdf. [Accessed January 25$^{\text{th}}$, 2011].

4 Laboratory 6: Cal Climber Part IV Model-Based Hill Climb in Statecharts Objectives and Checkout

Name: _____

Date: _____

Lab Section: _____

NOTE: The goal of this lab is to demonstrate your ability to solve a real-world problem using a state machine. The objectives below test different behaviors of your robot, and hence they must all be tested on the same version of your software. At checkoff, your robot will be tested on each objective.

1. **(_____ / 40)** Design a Statechart diagram that drives the simulated robot in a fixed orientation, while avoiding obstacles. From its initial starting point, it should eventually reach the ramp, at which point it must reorient towards the top. Your software must respond to collisions detected by momentary bump sensors on the front of the robot, and to cliffs detected by infrared distance sensors on the bottom of the robot. After collision avoidance, your robot should return to its original orientation. Use infrared cliff sensors, mechanical bump triggers, and mechanical wheel drop triggers to detect collisions or cliffs. Keep in mind that a collision or cliff event may occur *during* your collision avoidance algorithm, or while the robot is on the ramp.
2. **(_____ / 10)** Calibrate the accelerometer by running the calibration VI. The calibrated accelerometer should display values between $[-1g, 1g]$. You may change the exponential moving average coefficient to a value you feel is appropriate.
3. **(_____ / 50)** Program your robot to climb the inclined ramp, while avoiding cliffs and obstacles along the way. Keep in mind that a collision or cliff event may occur *during* your collision avoidance algorithm.
4. **(_____ / 0)** BONUS (up to +5): Program your robot to stop when it reaches the landing at the top of the inclined ramp.

5 Lab Writeup Prompts

In your lab writeup, please address the following questions:

1. A screenshot of your final Statechart diagram.
2. Open your Statechart diagram and use the "File, Generate Documentation" option to print extensive documentation of your Statechart. Do not print this documentation; instead, place the documentation in a .zip archive and e-mail to your lab instructor.
3. Did you use your previous calibration process, or did you use the provided calibration VIs? If you used the provided VIs, what changes did you notice from the calibration you had performed in the previous version of this lab?
4. Describe the architecture of the top-level target VI (**RT\Cal Climber.vi**). How many loops run concurrently? What is their period? Which is the highest priority, and which is the lowest, and why?
5. What did you like about this lab, and what would you change?

6 Laboratory 6: Cal Climber Part IV Model-Based Hill Climb in Statecharts Prelab Exercises

Name: _____

Date: _____

Lab Section: _____

1. Show that, if quantization error is neglected, that $\vec{a}_{\mathrm{g}}(t)$ can be exactly reconstructed from Eq. (3).