CSE4421: Lab 2

Burton Ma

Thu 27 Jan, 2011
Updated: Wed 09, Feb, 2011 with changes for OpenCV

1 Preliminaries

This lab teaches you some of the functions available to you on the A150 arm that are accessed through Minicom; you will probably need to use some of these functions for your project, and you may wish to try to modify the C serial port library and the Java arm controller (from Lab 1) to make these functions available to you without requiring Minicom.

This lab also introduces you to the Matlab interface for the A255 robot. The A255 API is somewhat different from that of the A150. A Matlab interface to the A150 is also available.

Finally, this lab shows you how to acquire images and video from the Firewire cameras available in the lab using OpenCV. The cameras may be useful for your projects.

2 Using RAPL2 and the A150 Arm

After you use Minicom to connect to the A150 arm, you can send commands to the arm using the RAPL2 language; the language is fully documented in the second half of the A150 binder. You should scan over the documentation before proceeding with this part of the lab.

Power up the A150 arm as you learned in Lab 1, but do not exit Minicom. There are numerous commands that you might find useful when doing your project. Here is a partial list (I recommend you learn how to use them in this order):

- LIMP and NOLIMP
- CLOSE, OPEN, and GRIP (do not use GRIP to hold onto an object)
- W0 and W2 (what distance units are being used?)
- JOG
- HERE
- APPRO and DEPART
- CPATH

3 Instructions to Use the A255 Arm

1. Log in to the workstation connected to the arm.
2. Start a console; in the console start the minicom program by entering the command `minicom`.
3. Find the teach pendant that plugs into the socket located in the upper-left corner of the controller. Plug the teach pendant into the controller (the arm will not power on without the pendant plugged in).
4. Power on the robot by pressing the power switch in the middle-left side of the control panel. Wait for it to finish its boot process.
5. Check the big red kill switch (kills power to the arm motors). If it is pressed in, unlock it by turning it.
6. Check the kill switch on the pendant. If it is pressed in, unlock it by turning it.
7. Press the arm power button located in the upper-right corner of the controller. You should hear a click and see a light on the button.
8. Check if the robot is in a suitable pose so that it can be homed (all of the joint markers should be approximately aligned). If it is not, manually rotate the joints by entering the *joint* command in Minicom (type *help joint* for documentation or see Chapter 3 of the *CROS and System Shell* manual).
9. Tell the robot to home itself by entering the command `home` in minicom. This will take some time.
10. Once the robot is correctly homed it is ready for use. Enter the command `ready` in Minicom.

3.1 Instructions to Turn Off the A255 Arm

1. If the robot joints are limped, unlimp them.
2. Press the kill switch; the robot has joint brakes that will lock the joints in place.
3. Turn off the main power on the controller.

3.2 Using the Application Shell

The user can interact with the A255 using the application shell (see the *Application Shell ASH* manual). To start the application shell type the following command into Minicom:

```
ash test
```

Once in the application shell, you can list all of the available commands by typing *help*, and you can get help on a particular command (say the *joint* command) by typing *help joint*. Of course, you can always refer to the *Application Shell ASH* manual. Like the A150, there are numerous commands that you may find useful if you choose to use the A255 for your project:

- LIMP and NOLIMP
- GRIP_CLOSE and GRIP_OPEN (not that WGRIP does not work properly on this arm)
- W0 and W2 (what distance units are being used?)
- WX, WY, and WZ
- TX, TY, and TZ
- ROLL and PITCH
- HERE
- MOVE
- APPRO and DEPART

3.3 Using the Matlab Controller

It is impossible (?) to write a program that runs in the ASH shell; however, a controller written in Matlab is available for your use:

```
/cs/dept/www/course_archive/2010-11/W/4421/src/crsarms.zip
```

(also available from the Source Code section of the course web pages). In the ZIP archive are demonstration Matlab scripts that show you how to use the controllers for the A150 and A255. The Matlab controllers implement greater functionality than the Java controller, and can easily (?) be extended to provide every (?) RAPL2 command (for the A150) and ASH command (for the A255).

There is an important difference between the Java controller and the Matlab controllers. The Java controller and simulator use the DH convention to specify the joint angles. The Matlab controllers use the robot's native convention for the joint angles; specifically, the joint angles for the joints 2, 3, and 4 (the shoulder, elbow, and first wrist joint) are specified relative to the horizon. In the ready position, the robot's convention is that the first four joint angles are waist 0°, shoulder 90°, elbow 0°, and wrist 0°; these are same values you should use when using the Matlab controller function `madeg`. The same joint angles will cause the arm to point stright up when using the Java controller.

4 Acquiring Video and Images

The robotics lab is equipped with 3 small blue Firewire cameras. You may want to use a camera for some sort of vision-based input or control of the robotic arms for your project.

You might consider using OpenCV for any vision-based task that you require for your project (although you may use any other API if you want). I have placed two copies of the book "Learning OpenCV: Computer Vision with the OpenCV Library" in the robotics lab. Please do not remove the books from the lab, and do not photocopy them.

The cameras, cables, and books are located in the black cabinets. Please put the equipment away when you are finished with it at the end of the day.

The cameras draw power from the Firewire port; all you need to do to use a camera is plug it in.

4.1 Using OpenCV to Acquire an Image

Chapter 2 of "Learning OpenCV" is a nice introduction to the OpenCV library. If you are interested in using a camera for your project, this would be a good time to read this chapter to familiarize yourself with OpenCV. The first two chapters are available here:

```
/cs/dept/www/course_archive/2010-11/W/4421/doc/LearningOpenCV_1_2.pdf
```

(also available from the Documentation section of the course web pages).

If you want to get started right away without reading the introduction, there is a small program that you can study and compile here:

```
/cs/dept/www/course_archive/2010-11/W/4421/src/testCamera.c
```

(also available from the Source Code section of the course web pages). The command to compile the program is given in the comments at the top of the source code file. See the end of this section for information on setting the `LD_LIBRARY_PATH` environment variable so that the test program will run.

Another source of OpenCV information is:

```
http://opencv.willowgarage.com/wiki/FullOpenCVWiki
```

This page is the Wiki for the OpenCV community. Find the link under Frequently Asked Questions that shows you how to capture and display images from a camera. Once you've edited and saved your program, you can compile it using the command (note that the backwards apostrophes ` are important!):

```
setenv LD_LIBRARY_PATH /cs/local/lib/gcc/i686-pc-linux-gnu/4.4.1:/cs/local/lib
gcc-4.4.1 `pkg-config --cflags --libs opencv` -o myapp myapp.c
```

You only need to set the `LD_LIBRARY_PATH` environment variable once in a terminal (but you must set it for every terminal you open). You can set the variable in your `.cshrc` configuration file, which will cause the variable to be set automatically every time you open a terminal.

4.2 Running Your OpenCV Program

If you have not already done so, you will need to set an environment variable before running your program:

```
setenv LD_LIBRARY_PATH /cs/local/lib/gcc/i686-pc-linux-gnu/4.4.1:/cs/local/lib
```

If the image is corrupted, or if there are errors connecting to the camera, then you will need to reset the camera and FireWire connection using the command:

```
reset_camera
```

Resetting the camera is often required the first time you run an OpenCV program after logging in, and if you change cameras.