EECS 222: Embedded System Modeling
Spring 2017

Assignment 1

Posted: April 3, 2017
Due: April 10, 2017 at 12pm (noon)

Topic: Setup and Introduction to Application Example

1. Setup:

A Linux account has been created for you on the EECS department servers. The login and password credentials are the same as for your UCInetID.

Any of the following hosts can be used as computing server:

```
crystalcove.eecs.uci.edu
zuma.eecs.uci.edu
bondi.eecs.uci.edu
laguna.eecs.uci.edu
```

In order to remotely connect to the server, you will need to use the secure shell protocol (SSH) via a client software terminal. Note that we will mostly use the command line interface, so a simple terminal program (e.g. **`Putty`** for Windows) will be sufficient. Occasionally, however, we will also need graphical tools with GUI for which you will need X client software (e.g. **`Xming`**). See the course web site resource page for pointers to suitable SSH clients.

Use your SSH terminal to connect to one of the servers and make yourself familiar with the available commands in the Linux environment. You may also configure your shell setup to have a convenient working and C/C++ programming environment.

For this course, you will need to be comfortable with editing text and source code files in a text editor (e.g. **`vi`**, **`emacs`**, or **`gedit`**). You will also need to be familiar with the GNU C/C++ compiler chain for building executable programs. If you are not familiar with these tools, study available online tutorials and other resources and practice C/C++ programming in Linux.

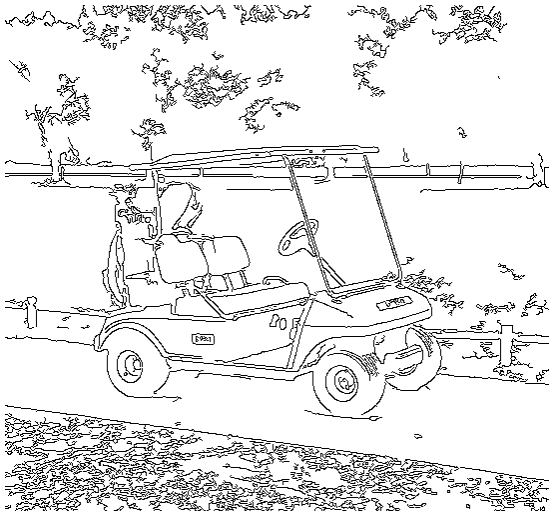
Since our modeling example is an image processing application, you will also need tools for handling digital images. There are many command-line tools available in Linux that allow you to convert and manipulate images. Most start with **`pnm`**, **`pbm`**, or **`pgm`**, depending on the file format they process. To enumerate them, you can type their prefix into your shell followed by a **`TAB`**. Documentation is available via the corresponding **`man`** pages.

In addition, graphical tools are available as well (if you have an X client running), which may be more convenient to use. To view images, you can use **`eog`** which supports most image types (including our pgm format). Finally, to manipulate images (e.g. if you want to use your own photos as a test case for our application), you can use **`gimp`**, a very powerful image editor in Linux.

2. Application Example

For the embedded system modeling project in this course, we will use a specific image processing application, namely an implementation of the *Canny Edge Detector* algorithm. The overall project goal is to design a suitable embedded system model of this application and describe it in a System-Level Description Language (SLDL). This embedded specification model will then not only be simulated for functional and timing validation, but also be refined for synthesis and implementation as an embedded System-on-Chip (SoC) suitable for use in a digital camera.

The Canny Edge Detector algorithm takes an input image, e.g. a digital photo, and calculates an output image that shows only the edges of the objects in the photo, as illustrated in the figure below. We will assume that this image processing is to be performed in real-time in a digital camera by a SoC that we design and develop.



Please refer to the following sources for more information on the Canny application:

http://en.wikipedia.org/wiki/Canny_edge_detector

http://marathon.csee.usf.edu/edge/edge_detection.html

You may refer to other web sites as well to study the Canny approach for edge detection in digital images.

3. Instructions

The purpose of this first assignment is for you to become familiar with the Canny algorithm and its application source code.

Step 1: Download the application source code

You can download the Canny application in C source code from the web site at ftp://figment.csee.usf.edu/pub/Edge_Comparison/source_code/canny.src.

Alternatively, you can copy the same source file and a suitable input image from our dedicated instructor account for this course:

```
mkdir hw
mkdir hw/hw1
cd hw/hw1
cp ~eecs222/public/canny.src .
cp ~eecs222/public/golfcart.pgm .
```

We will use this C reference implementation of the Canny edge detection algorithm as the starting point for our embedded design model. The golf cart image will serve as an initial input image.

Step 2: Test the given C code

Convert the downloaded **`canny.src`** file into a *single* ANSI-C file **`canny.c`**. Few adjustments will be necessary in order to cleanly compile the application on our Linux infrastructure with a modern compiler chain. Then you can compile and test the application, similar to the following:

```
vi canny.c
gcc canny.c -lm -o canny
./canny golfcart.pgm 0.6 0.3 0.8
eog golfcart.pgm_s_0.60_l_0.30_h_0.80.pgm
```

The generated output image should look similar to the one shown above.

Step 3: Study the application

Before we go and write a system-level model of this application, we need to study and understand it well. Examine the source code and its execution!

You should be able to answer questions like the following:

What are the main functions of the algorithm? Which functions are used for data input and for data output? Where does the actual computation occur? Which of the functions is the one with the most complexity? Can we expect the application to run in real-time as required by our overall goals? How much memory is

needed when the algorithm runs? Are there any obvious candidates for hardware acceleration? What should better be performed in software? …

Some of these questions are easy, some are harder. You don't need to have all the answers, but be prepared to discuss these issues in class.

Note that there is nothing to submit for this assignment at this time, but you will need the results in order to complete the following assignments.

--
Rainer Doemer (EH3217, x4-9007, doemer@uci.edu)