Embedded Systems Modeling and Design
ECPS 203
Fall 2021


# Assignment 1

**Posted:** September 27, 2021
**Due:** October 6, 2021 at 6pm

**Topic:** Setup, introduction to application example Canny Edge Detector

## 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 remote 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 use graphical tools with GUI for which you need X client software (e.g. **`Xming`**). See the resource page on the course web site for pointers to suitable SSH clients on your laptop.

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`**, **`nano`**, **`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.

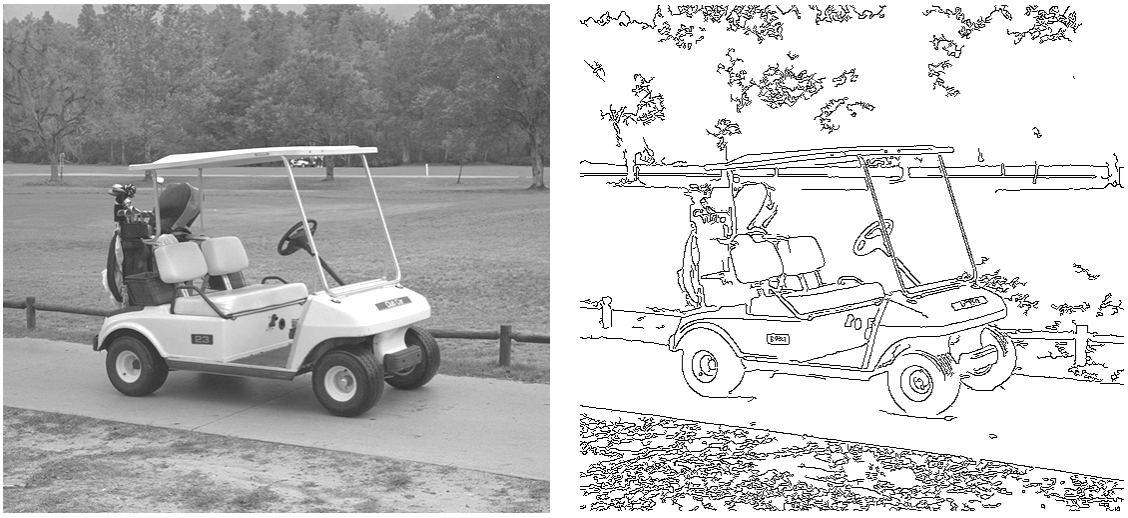
Since our modeling example is an image processing application, you will also need tools for handling digital images. There are many command-line tools available in Linux that allow you to convert and manipulate images. Most start with **`pnm`**, **`pbm`**, or **`pgm`**, depending on the file format they 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 the needed **`pgm`** and **`pnm`** formats). 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 design 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 the IEEE SystemC System-Level Description Language (SLDL). The created embedded specification model will then be simulated for functional validation and later also be refined for real-time implementation on an embedded platform with a digital video 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 connection with a digital camera. (If we are successful with our modeling, a follow-up course can then develop a real-time implementation of this application on an actual prototyping board.)



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

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

John Canny, “A Computational Approach to Edge Detection”, IEEE TPAMI, 1986. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf

You may refer also to other web sites in order to further 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 Edge Detector application in form of 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 ~ecps203/public/canny.src .
cp ~ecps203/public/golfcart.pgm .
```

We will use this C reference implementation of the Canny Edge Detector 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!

As an initial step in understanding the application, examine the main functions in the source code and create a corresponding function-call tree that shows which function calls which other function(s). Document your function-call tree in form of a simple text file named **`canny.txt`** and submit it for this assignment.

Towards the end of this course, we will also answer the following questions:

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. While we don't have all the answers at this time, we will have those (and many more) at the end of this course.

**4. Submission:**

For this first homework assignment, turn in the following deliverables:

```
canny.c
canny.txt
```

To submit these files, change your working directory to the parent directory of your **hw1** directory and run the **~ecps203/bin/turnin.sh** script. This command will locate the current assignment deliverables and allow you to submit them, as follows:

```
doemer@bondi.eecs.uci.edu: ~ecps203/bin/turnin.sh
=================================================
ECPS203 Fall 2021:
Assignment "hw1" submission for doemer
Due date: Wed Oct  6 18:00:00 2021
** Looking for files:
**   canny.c
**   canny.txt
=================================================
* Please confirm the following:                 *
* "Following the Academic Honesty Policy at UCI,   *
* I submit my own original work."               *
   Please type YES to confirm. y
=================================================
Submit canny.c [yes, no]? y
  File canny.c has been submitted
Submit canny.txt [yes, no]? y
  File canny.txt has been submitted
=================================================
  Summary:
=================================================
Submitted on Mon Sep 27  9:58:02 2021
You just submitted file(s):
  canny.c
  canny.txt
```

Note that you can use this **turnin.sh** script to submit your work at any time before the deadline, *but not after!* Since you can submit as many times as you want (newer submissions will simply overwrite older ones), it is highly recommended to submit early and even incomplete work, in order to avoid missing the deadline.

*The deadline is hard. Late submissions will not be considered!*

To double-check that your submitted files have been received, you can run the following script:

```
doemer@bondi.eecs.uci.edu: ~ecps203/bin/listfiles.py
============================================================
ECPS 203 Fall 2021: "hw1" listing for doemer
============================================================
Files submitted for assignment "hw1":
canny.c
canny.txt
```

For any technical questions, please consult with the TA in the discussion session or use the course discussion forum.

Footnote: Please note that the Linux commands in this document may or may not work when using copy-paste operations (due to different font encoding between the PDF and the shell).

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