Embedded Systems Modeling and Design
ECPS 203
Fall 2017

Assignment 7

Posted: November 15, 2017
Due: November 22, 2017 at 6pm

Topic: Performance measurement of the Canny Edge Detector on prototyping board

1. Setup:

This assignment continues the *Canny Edge Detector* project from the application source code produced in Assignment 4 (not 6!). While our SystemC model can be simulated on the Linux servers and provide relative timing measurements (Assignment 6), we will need to also obtain some absolute measurements on the target platform to estimate the expected real-time performance there.

Thus, we will take the application C++ model and measure its run-time on the Raspberry Pi prototyping board. These measurements will then serve as an absolute reference point for performance estimation of the actual embedded system we envision for the final implementation.

For submission purposes, we will use the same setup as before. Start by creating a new working directory on the server, so that you can properly submit your deliverables in the end. You should also use this directory for preparing the files needed on the board, as well as a safe storage space for any data you may need to backup from your prototyping board while running this Canny measurement experiment.

```
mkdir hw7
cd hw7
```

As usual, you may use your own model and video images or the provided solution from Assignment 4, as follows:

```
ln -s ~ecps203/public/video video
cp ~ecps203/public/CannyA4_ref.cpp Canny.cpp
```

However, for the actual experiment, you will now use the Raspberry Pi prototyping board that you have received at the beginning of the quarter as part of the MECPS program.

2. Compiling and running the Canny Edge Detector on the prototyping board

The following instructions outline how to instrument the Canny C++ application for real-time measurements and then run it on the Raspberry Pi board to observe the timing.

Step 1: Prepare your prototyping board

1) While not strictly required, we recommend that you make a backup of all the contents on your Raspberry Pi SD card. Thus, if anything goes wrong during this experiment, you have a safe point to return to.

2) Install the Raspbian software on the Raspberry Pi board. (If you have already installed Raspbian on your Raspberry Pi, you can skip this step.)

   a) Go the the offical website of Raspberry Pi, and click "DOWNLOADS"

   b) Download "NOOBS" as a zip file

   c) Unpack the "NOOBS" package onto your SD card

   d) Plug the SD card into your Raspberry Pi and connect the board to a monitor, a mouse and a keyboard. Power up the board.

   e) Select Raspbian as the operating system, and NOOBS will automatically install it for you

Step 2: Compile the Canny C++ application

Again, we will use as starting point the clean application source code of the Canny Edge Detector that you have prepared in Assignment 4.

3) Copy the source file `Canny.cpp` and the input images to your Raspberry Pi. You can do this either via a USB flash drive or by connecting your Raspberry Pi to the network.

4) Open the terminal. You can find it in the menu bar which is on top of your screen.

5) Change into the directory where your source file is stored. For example, if you put it in `project/hw7`, then go there:

```
cd ~/project/hw7
```

6) Compile the Canny application with the GNU compiler to ensure it works on your board

```
g++ -Wall canny.c –o canny
./canny
```

Step 3: Instrument the source code with timing measurement instructions

1) Include the `time.h` header file in your `canny.cpp`

```
#include <time.h>
```

2) In order to start the timer, place the following statement right before the function call to be measured:

```
clock_t start = clock();
```

3) In order to stop the timer, place the following statement right after the function call to be measured:

```
clock_t finish = clock();
```

4) Then, to calculate the run time of the measured function in seconds, you can use a calculation like this:

```
double totaltime = (double)(finish-start)/CLOCKS_PER_SEC;
```

Step 4: Note the delays of the major Canny functions

5) Finally print the measured delays to the screen, for example, but using the **`printf`** function

```
printf("function_name run time %f secs\n", totaltime);
```

6) Using the above instrumentation, measure the real-time delay of all the functions in the Canny algorithm that we have represented as separate modules in the SystemC model of the previous Assignment 6.

Note that all functions will report multiple measurements due to the processing of the video stream. However, for our estimation purposes we are interested in the average run-time per function call. Thus, calculate the arithmetic mean of these values and report them.

3. Submission:

For this assignment, submit the following deliverables:

`canny.cpp`	(the Canny C++ model with timing instrumentation)
`canny.txt`	(a brief text file with the table of measurement results)

The text file should briefly describe your efforts and any problems you encountered, as well as report your measured timing.

To submit your deliverables, change into the parent directory of your **`hw7`** directory and run the **`~ecps203/bin/turnin.sh`** script. As before, this command will locate the current assignment files and allow you to submit them.

Again, late submissions will not be considered!

To double-check that your submitted files have been received, you can run the **`~ecps203/bin/listfiles.py`** script.

For any technical questions, please consult with the TA in the lab or use the course message board.

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