National University of Singapore
School of Computing

CS2281 **Programming Assignment 2** Semester 3 04/05

Deadline

Mon May 10 17:00:00 GMT-8 2004

Learning Keywords

command line argument, file I/O, binary file, 2-D array, dynamic memory allocation, math library.

Your Task

In this question, you are required to implement a program that reads in an image in PGM format, perform edge detection on the input, and write the resulting image to `stdout`. Figure 1 shows an example input image (left) and the corresponding output (right).

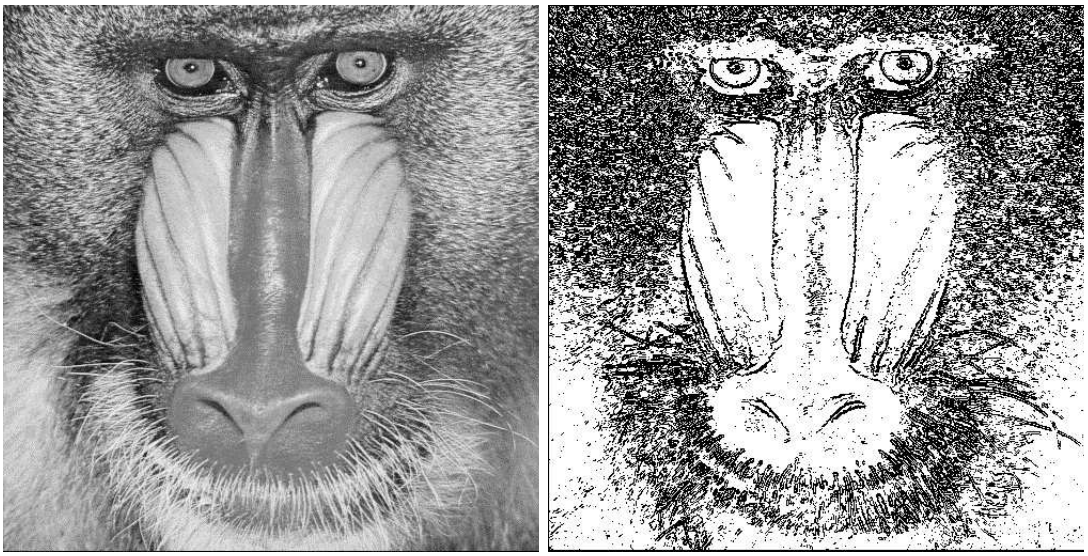

Figure 1: Example input and output image

Command Line Arguments

If your program is called without any argument, read the input PGM image from `stdin`. Otherwise, if a file name is passed to your program as command line argument, your program should read the input PGM image from the given file. In case more than one command line argument is passed to your program, your program should simply remind the user the usage of the program and quit.

PGM File format

PGM, or portable graymap format, is an image format for storing uncompressed grayscale image. There are several variations of PGM file format (`man pgm` for details.), but for the purpose of this assignment, we assume the following simplified format.

Each PGM file consists of the following:

- A magic number, which is the two characters "P5".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, w, formatted as ASCII characters in decimal.
- Whitespace.
- A height, h, again in ASCII decimal.
- Whitespace.
- The maximum gray value (maxval), again in ASCII decimal. You can assume this to be 255.
- Newline or other single whitespace character.
- A raster of $w \times h$ gray values, proceeding through the image in normal English reading order (i.e., left to right, top to bottom). Each gray value is a number from 0 through 255, with 0 being black and 255 being white. Each gray value is represented in pure binary by 1 byte.
- Lines that begin with a '#', before the maxval line, are comments and are ignored.

Your program must be able to ignore comments, and handle images of arbitrary size. If the magic number "P5" cannot be found at the beginning of the image, your program should output an error and quit. Otherwise, you can assume that the rest of the file is a valid PGM file. Your program should parse the header and read the pixel (raster) data into a dynamically allocated 2D array.

Edge Detection

We now describe how to perform edge detection and compute the output image. Refer to Figure 2. Given a pixel on the output image e, we compute two intermediate values x and y using the neighboring pixels of e from the *input* as follows:

$$x = (c + 2f + i) - (a + 2d + g)$$

$$y = (g + 2h + i) - (a + 2b + c)$$

a	d	g
b	e	h
c	f	i

Figure 2: To compute the value of output e, we make use the values of the surrounding pixels (a, b, c, d, f, g, h and i) from the input image.

The value of the output pixel at location e, is then computed using

$$e = \begin{cases} 0 & \text{if } \sqrt{x^2 + y^2} \geq \epsilon \\ 255 & \text{otherwise} \end{cases}$$

where ϵ is a threshold value which you must define as a constant using `#define`.

The above description does not take care the cases where e lies at the border of the image. For simplicity, you can simply set of the value of the border pixel to 255.

Submission Requirement

Make sure you have read the submission instruction document posted on CS2281 website. For this assignment, create a subdirectory under `$HOME/CS2281_LABs/` called `a2` and put your encrypted files under the subdirectory. You must name your program `edge.c` and include your name as a comment in the *first* line of the file. Submit your `Makefile` as well. I will access your homework using pathname `$HOME/CS2281_LABs/a2/edge.c.pgp` and `$HOME/CS2281_LABs/a2/Makefile.pgp`. It is your responsibility to make sure that the filenames are correct and permissions are set properly according to the instructions given.

Additional Tips

- To compute square root, use the function `sqrt` from math library. Not only you should include the header file `math.h` to access the declaration for this function, you need to link with the math library (`/usr/lib/libm.a` on `sunfire`) as well using:

  ```
  gcc edge.c -lm
  ```

The option `-l` tells gcc to link with a library, and `m` specifies the name of the library. (So, to link with `libbanana.a`, use `-lbanana`)

- A good value for ϵ is 128. You can try different values for ϵ.
- Break down your program into different logical units and code each one as a seperate function. For example, one function to open file, one function to skip comments etc.
- Sample PGM images can be found at CS2281 website. (Downlaod Here)
- You can view PGM images on `sunfire` using `xv`. (You need to run X first.)
- The edge detection algorithm above will not work well if the input image contains noise. You can use the command `pnmsmooth` (available on `sunfire`) to smooth the image first before passing it to your program. For example, assuming your executable is called `edge`,

  ```
  pnmsmooth input.pgm | edge > output.pgm
  ```

- You can run your program on JPEG images, by first converting the JPEG files to PGM, passing the output to your program, and coverting output from your program back to JPEG.

  ```
  djpeg -grayscale DCS00130.jpg | edge | cjpeg > output.jpg
  ```

  This example embodies two important UNIX philosophies, the Rule of Modularity and Rule of Composition (Read *The Art of UNIX Programming* for details).