SE 2S03 – Assignment 4

Ned Nedialkov

16 November

Due date: 30 November, in class

1 Introduction

This assignment is about performing simple noise reduction on images.

1.1 PPM

The Portable Pixmap Format (PPM) uses ASCII encoding of pixels for image files; for details, see https://en.wikipedia.org/wiki/Netpbm_format#PPM_example. Here we shall use the P3 encoding.

You can generate a PPM file from a JPG file using

```
convert  -compress none file.jpg  file.ppm
```

where `convert` is from ImageMagic, https://www.imagemagick.org/script/index.php.

On Linux, you can use `gimp` to view a PPM file. On Mac OS X, simply use `open`.

1.2 Moving filters

1.2.1 Mean filter

A simple filter for noise reduction in image processing is to replace a pixel by the average of the neighbouring pixels in a "sliding" window. For example, with a 3×3 window and the numbers on the left,

45	4	255
78	124	56
1	0	34

$\Rightarrow$

×	×	×
×	66	×
×	×	×

we obtain the center of the window on the right as (rounded to the nearest integer)

$$(45 + 4 + 255 + 78 + 124 + 56 + 1 + 0 + 34)/9 = 66.$$

Such a window goes through each entry and replaces it by the mean of the entries in the window, where the entry in the middle is included in computing the average.

1.2.2 Median filter

Another filter is to replace a pixel by the median of the pixels in the window. For details, see https://en.wikipedia.org/wiki/Median_filter.

1.3 RGB encoding

In an RGB encoding, we take the average (or median) over each of the colors red, green, and blue independently. That is, the average of all red, all green, and all blue pixels in the window.

2 Programming (20 points)

2.1 Reading, processing, and writing a PPM file

Implement the bodies of three functions specified in file filter.h; see Figure 1. Use this file and do not change it.

2.2 Main program

Write a main program that takes as input (from the command line) the following parameters:

- name of input file
- name of output file
- size of sliding window
- type of filter

You must have a `makefile` such that when `make` is typed, an executable with name `denoise` is created in the current directory.

Your program must run as

```
./denoise input.ppm output.ppm N F
```

where

- `input.ppm` is the name of the input file
- `output.ppm` is the name of the output file
- `N` specifies the size of the window

```
#ifndef INCLUDED_FILTER_H
#define INCLUDED_FILTER_H

/* Type of filtering */
typedef enum { MEAN, MEDIAN } filter;

/* RGB values */
typedef struct { unsigned char r, g, b; } RGB;

/* readPPM reads a PPM image from a file.
   Input: file is a pointer to a character string specifying a file name.
   Output:
     *width stores the width of the image
     *height stores its height
     *max stores the largest RGB value in the image

   readPPM returns a pointer to an array of size (*width)*(*height)
   storing the RGB values of the image.
*/
RGB *readPPM(const char *file, int *width, int *height, int *max);

/* writePPM writes an image into a file.
   Input:
      file   name of the file
      width  width of the image
      height height of the image
      max    largest RGB value
      image  pointer to an array of size width*height with RGB values
*/
void writePPM(const char *file, int width, int height, int max,
              const RGB *image);

/* denoiseImage applies filtering to an image.
   Input:
      width  width of the image
      height height of the image
      max    largest RGB value
      image  pointer to an array of size width*height with RGB values
      n      size of filtering window
      f      type of filtering: MEAN or MEDIAN

   denoiseImage returns a pointer to an arrary of size width*height
      containing the new image.
*/
RGB *denoiseImage(int width, int height, const RGB *image, int n, filter f
   );

#endif
```

Figure 1: File `filter.h`

- `F` is the type of filtering and can have a value `A` meaning mean filter, or a value `M` meaning median filter

For example,

```
./denoise input.ppm output.ppm 3 A
```

would apply a 3×3 mean filter on `input.ppm` and produce `output.ppm`.

You can test your program on this small file http://www.cas.mcmaster.ca/~nedialk/COURSES/2s03/private/ein.ppm. However, your program should work with any file. Figure 2 shows the result of my smoothing on the file `ein.ppm`.

Then try your program on this image http://www.cas.mcmaster.ca/~nedialk/COURSES/2s03/private/lux.ppm

Generate output similar to this one: e.g. calling

```
./denoise lux.ppm lux11A.ppm 11 A
```

produces

```
Reading file lux.ppm
***    lux.ppm read    in 5.7e+00 seconds

Processing 3672 x 4896 image using 11 x 11 window and mean filter...
***    image processed in 3.8e+00 seconds

Writing file lux11A.ppm
*** lux11A.ppm written in 4.9e+00 seconds
```

and calling

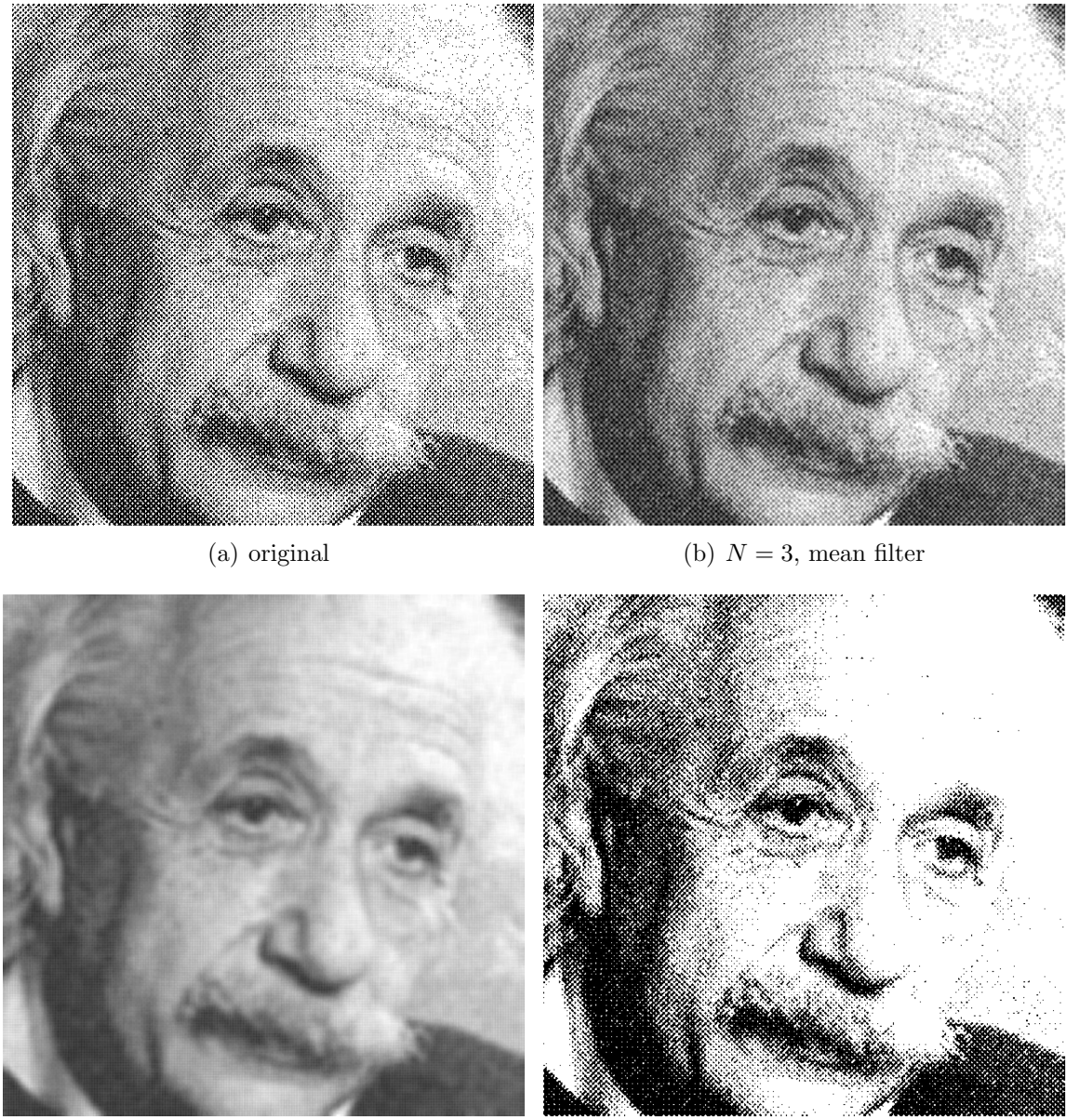
```
./denoise lux.ppm lux11M.ppm 11 M
```

produces

```
Reading file lux.ppm
***    lux.ppm read    in 5.7e+00 seconds

Processing 3672 x 4896 image using 11 x 11 window and median filter...
***    image processed in 5.1e+02 seconds

Writing file lux11M.ppm
*** lux11M.ppm written in 4.9e+00 seconds
```



(a) original

(b) $N = 3$, mean filter

(c) $N = 11$, mean filter

(d) $N = 5$, median filter

Figure 2: Results of smoothing of the original in (a).

3 Timing (2 points)

Produce timing results for each of the calls

```
./denoise ein.ppm ein3A.ppm   3 A
./denoise ein.ppm ein11A.ppm 11 A
./denoise ein.ppm ein5M.ppm   5 M
./denoise lux.ppm lux11A.ppm 11 A
./denoise lux.ppm lux11M.ppm 11 M
```

and complete the following table

		CPU time	
run	reading	processing	writing
`ein3A`			
`ein11A`			
`ein5M`			
`lux11A`			
`lux11M`			

4 Profiling (5 points)

a. (2 points) Profile the execution of

```
./denoise lux.ppm lux11A.ppm 11 A
```

Report the lines of code that contribute the most to the total execution time and total to about 80% of the execution time of your program.

You can take the output of `gprof` and highlight those lines in your hard copy submission.

b. (3 points) Can you explain why they take so much time?

5 Bonus (up to 5 points)

Can you optimize your code such that it shows much better timing results than the output on page 4? This output is produced on `mills` with optimization option `-O2`. (My code is not optimized though.)

If you manage to reduce the time substantially, discuss how you have done it.

Report your timing results on `mills` with optimization option `-O2` for

```
./denoise lux.ppm lux11A.ppm 11 A
./denoise lux.ppm lux11M.ppm 11 M
```

6 What to submit

- Hardcopy of all your code.
- Your code using subversion to svn under directory `A4`.
- The resulting files of

```
./denoise ein.ppm ein3A.ppm    3 A
./denoise ein.ppm ein11A.ppm  11 A
./denoise ein.ppm ein5M.ppm    5 M
./denoise lux.ppm lux11M.ppm  11 M
```

  to both subversion and as images in the hardcopy (no need to use a color printer).