Assignment 4 - Software I, Spring 2003 (0368-2157-9,0368-2157-12)

`http://www.cs.tau.ac.il/~efif/courses/software1`

Due: April 29, 2003

Before starting to answer the questions, please read very carefully the "Submission Guidelines"[1]. Make sure your programs detect invalid input data, and print out appropriate error messages. Do not add "friendly" messages to your programs, as they are tested automatically by other programs.

Ex 4 photo

In this assignment you are asked to implement two methods for scaling down two dimensional images, using the photo program shown in class as your framework. Before you continue, please download the source file of the photo program from the web practice-page at
`http://www.cs.tau.ac.il/~efif/courses/software1/code/photo/photo.c` , and familiarize yourself with the code. Then, add the command-line options below.

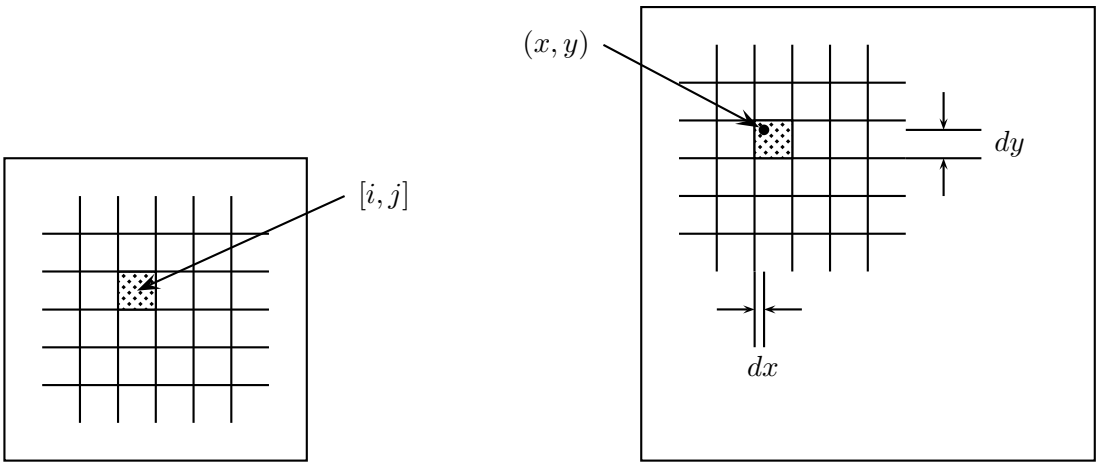
The photo program reads an image in ASCII Portable Pixmap File (PPM) format as input. In this format each pixel is a triplet of red, green, and blue samples one character each, in that order. The program opens a window using the glut library, and displays the image on the window. You need to enhance the program to perform the required operation non interactively based on various command line options. For your convenience, you may enhance the program to perform the same operation interactively, as a response to various user key-strokes.

Let w_s and h_s be the width and the height of the source image respectively, and w_d and h_d be the width and the height of the destination image respectively. Let $C_s[i,j]$ and $C_d[i,j]$ denote the pixel colours of the source image and destination image respectively at position $[i,j]$. The formulae below for $C_d[i,j]$ must be applied to each of the red, green, and blue components.

Let (x,y) be the non-integer position in the source image that corresponds to the position $[i,j]$ in the destination image, and let dx and dy be the fractional part of x and y.

$$x = i \times w_s/w_d \qquad dx = x - \lfloor x \rfloor$$
$$y = j \times h_s/h_d \qquad dy = y - \lfloor y \rfloor$$

[1] `http://www.cs.tau.ac.il/~efif/courses/software1`



Command Line Options

-s factor

set the scale factor. The default is 1.0.

-m method

select the scale method. The accepted methods are:

nearest

this indicates the nearest-neighbor method, which is the default. The nearest-neighbor method is the most simple method to resize an image. This method finds the closest corresponding pixel in the source image for each pixel in the destination image.

$$C_d[i, j] = C_s[\lfloor x \rfloor, \lfloor y \rfloor]$$

cubic

this indicates the cubic B-Spline method. This method estimates the colour of a pixel in the destination image by an average of the colour of the 16 pixels surrounding the closest corresponding pixel in the source image.

$$C_d[i, j] = \sum_{m=-1}^{2} \sum_{n=-1}^{2} C_s[\lfloor x \rfloor + m, \lfloor y \rfloor + n] R(m - dx) R(n - dy)$$

where

$$R(x) = \frac{1}{6}[P(x+2)^3 - 4P(x+1)^3 + 6P(x)^3 - 4P(x-1)^3]$$

$$P(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Assume that the border of the source image is replicated as many times as necessary, when computing the pixels near the border of the destination image. That is, $C_s[i, j] = C_s[0, j]$ for $i < 0$, and $C_s[i, j] = C_s[i, 0]$ for $j < 0$.

-t

test the scale down operation.

If this option is not indicated, the program displays the input image, and awaits for instructions from the user in form of key strokes. When this is entered by the user however, the program executes in non-interactive mode. First, the image is read. Then, it is transformed according to the selected scale method. Finally, its signature is calculated as explained below, and printed out in hexadecimal format. The program exits immediately after. This option will be used by the automatic testing script. The signature is simply the sum of all colour components of all pixels calculated with **unsigned int**.

Examples

Input:

```
photo -s 2.0 -m cubic -t photo.ppm
```

Output:

```
0x14e4734
```

This command tests the cubic B-spline scale method on the input image `photo.ppm`. The image is scaled down by a factor of 2.0.

Input:

```
photo photo.ppm
```

This command displays the input image `photo.ppm`.

Additional instructions

- Use `double` type to represent all real numbers in the program.
- For your convenience add the option below to the `keyboard` function, to allow the user to transform the image interactively.

  's' - scale down the image using the selected scale method.

- The program must be linked with the glut library. Add the command line option `-lglut` to the compilation command line:

  ```
  gcc -Wall -lglut photo.c -o photo
  ```

- Make sure that every allocated memory block is deallocated when the program exists, or better yet, when the block is not needed any longer.
- If you want to practice on windows and compile with Visual Studio, you can download the cross-platform source-code file from `http://www.cs.tau.ac.il/~efif/courses/software1/code/photo/cp_photo.c`. Beware, the signatures produced by an executable compiled on windows with Visual Studio may deviate from those produced by an executable compiled with gcc on Linux.

- You can use IrfanView to convert more images to PPM format . IrfanView is free and can be downloaded from `http://www.irfanview.com/`.

Good Luck!

More Information on the Submission

Files Name

The files for the exercises should be located under `~/software1/assign4`, and their names should match the name of the exercise. For example, the `C` source file and corresponding executable for exercise 4, namely photo, should be `~/software1/assign4/photo.c` and `~/software1/assign4/photo` respectively. Note that names are case sensitive (i.e. `ex1.C` is different than `ex1.c`).

Giving Permission to the Files

Before submitting the solution set, please give permission to the files by executing the following command:

```
chmod 705 ~ ~/software1 ~/software1/assign4 ~/software1/assign4/*
```