CSCI 3280 Introduction to Multimedia Systems

Spring 2020, Assignment 1 - Halftone

Due Date : Feb. 07, 2020 (11:59pm)

Submission via blackboard.cuhk.edu.hk

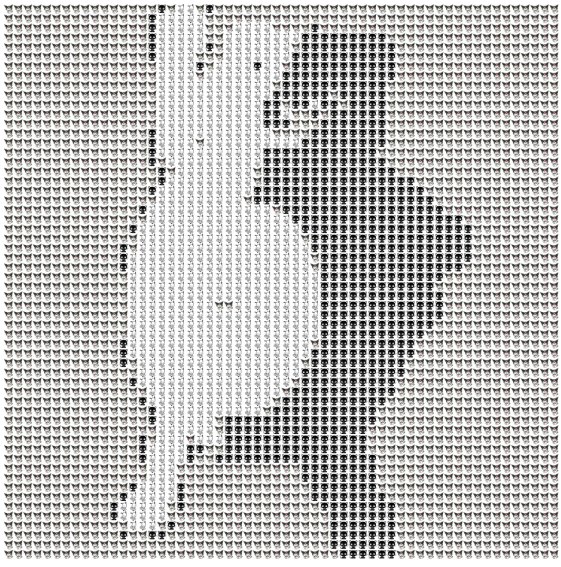
Late submission penalty : 10% deduction per day

PLAGIARISM penalty : Whole Course Failed

Introduction

The printing media, like newspapers or magazines, often uses images for illustration purposes. Given a common-quality newspaper, if zoom-in the image, we will see many dots composing the image. Image halftoning is one of the techniques to convert a gray or color image to the dot-pattern image.

In this assignment, you will be asked to implement a simple halftoning program. Different from traditional image halftoning, we generate results with given images instead of dot patterns.


into


with


Standard Requirements (100 point)

1. Program must be coded in ANSI C/C++ and uses standard libraries only
2. The compiled program must run in Windows 10 command prompt as a console program and accepts source bitmap (.bmp format) with the following syntax and save generated images to the current directory.

   ```
   C:\> halftone <input.bmp> <size1> <size2>
   ```

   halftone is your program executable
   e.g. **halftone monokuma.bmp** creates an halftone for monokuma.bmp
   <input.bmp> is the path name to the source bitmap
   <size1> and <size2> is the size of content image and patch images
   e.g. **halftone monokuma.bmp 64 16** creates an image with 64*64 patches whose size is 16*16.
3. Code fragment for read/write/resize .bmp format is provided in the skeleton code.
   Hint: Please read the code carefully and try to use the functions.
4. You are required to **submit source code only**. We will use Visual Studio 2015 C++ compiler and have your program compiled via visual studio command prompt (Tools→Command Prompt) and the following command line (***Please make sure your source code gets compiled well with it***).

   ```
   C:\> cl halftone.cpp bmp.cpp
   ```

5. Test bitmaps are included with the skeleton code for testing.

Implementation Details

Your program should process the RGB image similarly as the following steps:

1. **Reading RGB image** - Read in the .bmp as a RGB image, each pixel has R, G and B channels and each channel uses ‘unsigned char’ (i.e. one single byte) as their storage unit.
2. **Resize RGB image -** Resize the image with provided size. (Use the function in bmp library)
3. **Obtain Luma (Brightness)** - Convert the RGB image into a grayscale one using the following formula (based on CCIR 601 luma formula)

   ```
   Y’ = 0.299 * R + 0.587 * G + 0.114 * B
   ```

 into 

4. **Quantization** - Quantize the grayscale values into 3 levels (i.e. 0 to 2)
5. **Image Generation** - Map each level to an element from the given image patches .

Bonus Part (10 point)

You are encouraged to implement the following enhancement plus some features that you find interesting on top of the standard requirements. Please put the program with standard requirements plus enhanced features into a separate standalone source file and name it **halftone_bonus.cpp**

- Able to resize the image to specific width and height
- Quantize based on statistic information.
- Impress us and be creative!

Submission

We expect the following files zipped into a file named by your CWEM (e.g. s1234567890.zip) and have it uploaded to the Blackboard by due date: **Feb. 07, 2020 (11:59pm)**

- **README.txt** (Tell us anything that we should pay attention to, especially about the bonus part)
- **bmp.h & bmp.cpp** *(No need to change)*
- **halftone.cpp** (Standard requirement source code)
- **halftone_bonus.cpp** (OPTIONAL, Standard + bonus part source code)