

COMP7503
Multimedia Technologies

Programming Assignment

Deadline: 7 Dec 2017, before 7:00pm

Preamble:

This assignment is composed of two parts: a) Programming Part, and b) Written Part. The purpose of this assignment is to get you familiar with the concepts of 1) Color Space Conversion; and 2) Lossless Image Compression, through writing computer programs to process some real images. A program template based on Microsoft Visual Studio 2012 development environment will be provided. User interface has been developed so that your focus can be put only on algorithmic aspects.

The programming part requires you to implement a color space conversion algorithm, which converts color between RGB and YUV space, and investigate compression efficiency difference between the two color space. The written part, on the other hand, requires you to write a report, in which you explain in details your implemented algorithms in the programming part.







Please note that this is a group programming assignment. We expect two students working together to complete this assignment.

Overview:

A program template is first provided to you. Based on this template, you should implement a matched pair of **RGB to YUV** and **YUV to RGB** color space conversion algorithm, and also a matched pair of **compression** and **decompression** algorithm.

Within the template, you should regard a color image as arrays of pixels. Each pixel is being represented by 3 color components, namely Red, Green and Blue components. Each pixel is represented in 24 bits, with 8 bits for each color component.

On receiving the program template, you should be able to compile the program under Visual Studio 2012 without writing any code. The program template will take care of all the issues related to user interface, windows redraw and file read/write.

When you run the program, you can load image files in BMP, JPG, or PNG (without alpha channel) formats through the menu ("File->Open") or by pressing the button . In this assignment, you will be implementing algorithms for color space conversion and compression that can be triggered by the press , , , , and  buttons.

Programming Part:

For the programming part, you only need to focus on four files, namely:

1. **AppConvert.h** (under VPT/Header Files/App)
2. **AppConvert.cpp** (under VPT/Source Files/App)
3. **AppCompress.h** (under VPT/Header Files/App)
4. **AppCompress.cpp** (under VPT/Source Files/App)

You may want to have a look at the following functions in the AppConvert.cpp first.




```
unsigned char *CAppConvert::RGBtoYUV(unsigned char *pRGB, unsigned char *pYUV)
```

```
unsigned char *CAppConvert::YUVtoRGB(unsigned char *pYUV, unsigned char *pRGB)
```

pRGB and pYUV are pointers to image buffers for RGB color space and YUV color space. For pRGB, the R, G, B component (in 8-bit format) of a pixel at the location (i, j) can be addressed from this pointer as follows

```
B: pRGB[(i + j * width) * 3 + 0] ;  
G: pRGB[(i + j * width) * 3 + 1] ;  
R: pRGB[(i + j * width) * 3 + 2] ;
```

These two integers that would give you the width and height of a given loaded image is stored in the **width** and **height** variables.

As discussed before, you should be able to compile the program under Visual Studio 2012 without writing any code. You'll then able to run the program by pressing F5 key in Visual studio. You can then load some sample image files provided in the template through the menu ("File->Open") or by pressing the button . After successful loading the image, you can press the button  to bring up the console LOG window for seeing custom messages printed from the program. To test your color space algorithm implementation, you can press the  button, and two new windows "YUV444" and "YUV420" should popup. You'll be able to see the the YUV 444 format image, as well as a YUV 420 format image windows as shown in Figure 1 below. If you can see images that are almost identical to the original image in these two windows, you are most likely having a correct implementation of the color space conversion algorithm.

YUV 444 image format means that each each 2x2 patch pixel will be having 4 Y components, 4 U components, and 4 V components. YUV 420 format, on the other hands, mean that each 2x2 pixel will be having 4Y, 1U, and 1V component. In other words, in YUV 420 format, each pixel will be having its own Y component, but 2x2 pixels will be sharing the same U and V components.

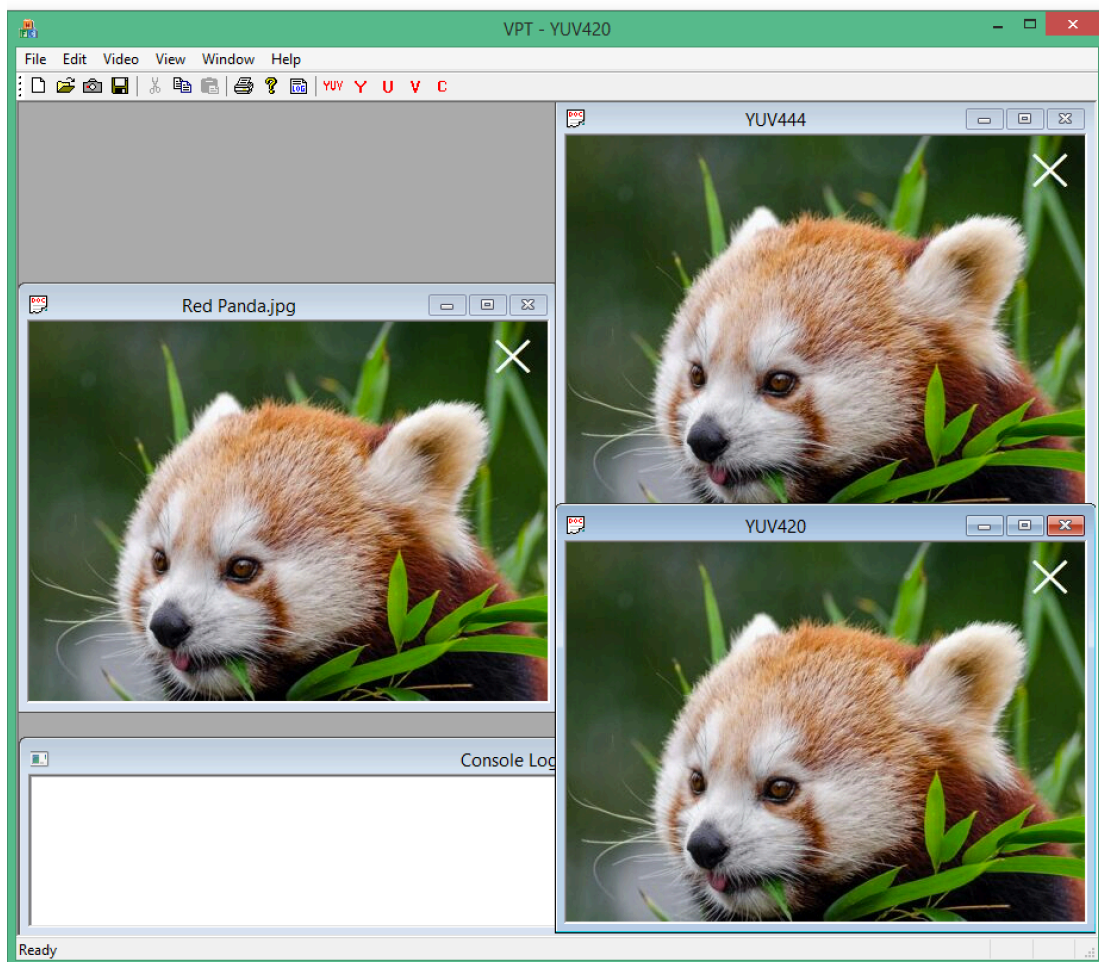






Figure 1: Sample Processing Results

First Programming Task

Your first task of this assignment is to complete the implementation of the following functions in the “*AppConvert.cpp*” file to realise the conversion between RGB and YUV colour image.

unsigned char *CAppConvert::RGBtoYUV(unsigned char *pRGB, unsigned char *pYUV)

unsigned char *CAppConvert::YUVtoRGB(unsigned char *pYUV, unsigned char *pRGB)


Upon successful completion of these functions, you should be able to see meaningful YUV 444 and YUV 420 images in after you press the  button. You may further try pressing the , , , buttons and they should trigger a new window showing the Y-, U-, and V-component images.

Second Programming Task

Your second task of this assignment is to complete the implementation of the following functions in the “*AppCompress.cpp*” file to realise the a lossless compression and decompression algorithms.

```
unsigned char *CAppCompress::Compress(int &cDataSize) ;
```

```
void CAppCompress::Decompress(unsigned char *compressedData, int  
cDataSize, unsigned char *uncompressedData) ;
```

To test the your compression and decompression implementation, you can press the button , and the window “Lossless Decompressed Image” will pop up and you can verify the correctness of your implementation. You may also see in the “Console Log” window for the performance of your compression algorithms, and it shall also alert you if your decoded image is not identical to the original image.

Programming Task Remarks:

Please note that you should keep the following function definition and parameter list unchanged.

```
unsigned char *CAppConvert::RGBtoYUV(unsigned char *pRGB, unsigned  
char *pYUV)
```

```
unsigned char *CAppConvert::YUVtoRGB(unsigned char *pYUV, unsigned  
char *pRGB)
```

```
unsigned char *CAppCompress::Compress(int &cDataSize) ;
```

```
void CAppCompress::Decompress(unsigned char *compressedData, int  
cDataSize, unsigned char *uncompressedData) ;
```

Apart from these functions, you cannot do modification to any other code anywhere else. However, you may define helper functions of your own for your algorithms implementations, but these should be isolated functions that can only be referenced or called within the four functions listed above. Any helper functions defined must be contained in the following files only:




1. *AppConvert.h* (under VPT/Header Files/App)
2. *AppConvert.cpp* (under VPT/Source Files/App)
3. *AppCompress.h* (under VPT/Header Files/App)
4. *AppCompress.cpp* (under VPT/Source Files/App)

Written Part:


You are required to write a report and describe in details the algorithms you have implemented in the programming part.

Besides, you'll also need to do the following experiments.

Experiment 1 procedure:

1. Load an image ;
2. Do a compression on the image, and note the compression ratio ;
3. Get mouse focus to the original image window again;
4. Click  button, and try do a compression on the Y-component image, and note the compression ratio ;
5. Repeat step 4 for  and  buttons.
6. Comment whether converting RGB image into YUV image would be more efficient for image compression ;

Experiment 2 procedure:

1. Load an sample image under the “Sample Images” folder;
2. Click on the  button;
3. Inspect the images YUV444 and YUV420, and note any difference between the two;
4. Repeat step 1 to 3 until you test all the Sample Images;
5. Comment whether YUV420 images would impact image quality;

In your report, you should document the findings from the two experiments above.

Assignment Submission:

Assignment must be submitted through the moodle course web. The following files have to be submitted as a one single zip file:

1. *AppQuantize.h*
2. *AppQuantize.cpp*
3. *AppCompress.h*
4. *AppCompress.cpp*
5. *Your written report*

Important Notes:

1. Late submission is subjected to the penalty of 5% deduction per day.
2. Submission late for more than one week will not be entertained.
3. Programs that cannot be compiled will NOT be graded.
4. Plagiarism will be heavily penalized.

=== End ===